PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 4 : G06F 9/44	A2	**(11) International Publication Number:** WO 88/ 07719 **(43) International Publication Date:** 6 October 1988 (06.10.88)

(21) International Application Number: PCT/US88/01081

(22) International Filing Date: 31 March 1988 (31.03.88)

(31) Priority Application Number: 033,340

(32) Priority Date: 31 March 1987 (31.03.87)

(33) Priority Country: US

(71) Applicant: AIMTECH CORPORATION [US/US]; 77 Northeastern Boulevard, Nashua, NH 03062 (US).

(72) Inventor: OLAPURATH, John, C. ; Six Woodfield Street, Nashua, NH 03062 (US).

(74) Agents: SMITH, James, M. et al.; Hamilton, Brook, Smith & Reynolds, Two Militia Drive, Lexington, MA 02173 (US).

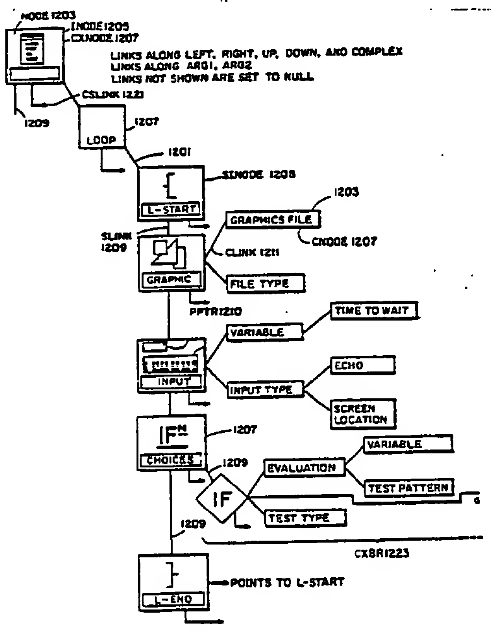
(81) Designated States: AT (European patent), BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: APPARATUS FOR ICONOGRAPHICALLY REPRESENTING AND EXECUTING A PROGRAM

(57) Abstract

A visual programming system used in a digital computer system which includes memory and a graphics display terminal. The program produced by the visual programming system is represented by a program structure in memory in which nodes represent program steps and links between the nodes the order of execution of the steps. The program represented by the program structure is executed by an interpreter component of the visual programming system. A display component of the system interprets the program structure to produce a display on the graphics terminal which represents the program as a structure of interconnected icons. The icons represent program steps and their interconnections specify the order of execution. Editing components of the system permit a program author to modify the program by manipulating icons representing the steps. The editing components include a structure editor which permits an author to add icons from an icon library to the icon structure and to move icons and copy icons already in the structure and a content editor which permits the author to add user-defined content to the program step represented by an icon. The author can further define complex icons, add them to the icon library, and use the complex icons in the program structure.



PROGRAM STRUCTURE (PSTRUC) 1201

APPARATUS FOR ICONOGRAPHICALLY REPRESENTING AND EXECUTING A PROGRAM

Background of the Invention

1. Field of the Invention

The present invention relates to computer systems and more particularly to systems for writing, displaying, and executing a program in a computer system.

2. Description of Prior Art

Efforts have been made since graphic display devices became available for use with computer systems to develop visual programming systems in which programs are written and modified by manipulating objects displayed on the graphics display device. In many such systems, the components of a program are represented by icons contained in a library and a user of the system writes a program by copying icons out of the library and linking them to other icons.

n example of such a visual programming system is Pict, described in Ephraim P. Glinert and Steven L. Tanimoto, "Pict: An Interactive Graphical Programming Environment", IEEE Computer, November 1984, pp. 7-25. The Pict system was developed to teach programming, and it provides the equivalent in graphic form of a high-level procedural language such as Pascal. At the beginning of a programming effort, the user has a library containing icons for primitive programming operations such as computation and assignment, branching, and looping. The user defines an icon representing a procedure, fetches primitive icons from the library, and connects them as required to perform the operation specified by the procedure. The resulting display resembles a classic flowchart. For example, each procedure has a single entrance and a single exit, and consequently, each branch must be connected either directly or indirectly to the exit. When the procedure definition is complete, the icon representing the newly-defined procedure becomes part of the library and can be used in defining other procedures.

When used in contexts other than the teaching of programming, Pict has the same difficulty as the classic high-level programming languages: the distance between the problem the programmer is trying to solve and the programming primitives he has available to him is so great that any but a trivial program requires a complex hierarchy of procedures. The effort and skill involved in developing the hierarchy and writing the procedures effectively bars non-programmers from using Pict to solve their problems. Other difficulties with Pict involve the degree of complexity of the display, the need to design an icon for each procedure, and the need to remember what each new icon stands for. Of course, all of these difficulties increase as the size of the program increases.

What is required if visual programming is to be more than a teaching tool or a laboratory curiosity is a visual programming system which may be easily used by the non-programmer in the solution of the problems relevant to the non-programmer's area of expertise. Such a system is provided by the present invention.

Summary of the Invention

The apparatus of the present invention is used in a computer system which includes memory and a display terminal. The apparatus includes a program structure in memory which represents a program to be executed in the computer system. The program structure represents the program by means of linked step nodes. The step nodes specify the steps in the program and the links between them specify the order in which the steps are executed. A display program executed by the computer system interprets the program structure to produce a representation of the program structure on the display terminal. In the respresentation, icons represent the step nodes and interconnections between the icons represent the links. When the program represented by the program structure is to be executed, an execution program executed by the computer system executes the steps represented by the step nodes in the order specified by the links.

Another aspect of the invention include an editing program which responds to editing commands. The editing program modifies the program structure and the result of the modification is immediately reflected in the displayed representation of the program structure. In a preferred embodiment, certain of the step nodes serve as root nodes for binary trees of content nodes which contain values used in the execution of the root step nodes. The step node and its content node are separably editable. When a user is editing the content node, the editing program is specific to the type of the step node. When a step node has been given content, it changes its appearance to indicate that fact. The program structure may be executed even though the step nodes are lacking some or all of their content.

The invention further includes an icon library used by the editing program. The icon library includes built-in simple icons representing step nodes. A build command permits a user of the system to copy a step node into the program structure by specifying the icon representing the step node in the library and the location in the displayed representation at which it is to be added.

A feature of the program structure is an unconditional branch introduced by a branch step node which is linked both to the first node in the branch and to the step node following the branch step node. The execution program executes the step nodes in the branch until the execution of the branch is complete and then executes the step node following the branch step node. A sequence of icons in the displayed representation may be defined as a branch and a complex icon representing the branch added to the icon library. The branch may be added to the program structure by specifying the complex icon representing the branch in the library and the location at which the branch is to be added. Complex icons in a preferred embodiment have an appearance different from other icons. Another feature of the program structure is a load step node which specifies a second program structure. When the execution program executes the load step node, it executes the second program structure and then continues execution with the step node following the load step node.

Further features of the invention include an arrow icon which is displayed to indicate the direction of a branch and the kind of step represented by the first node in the branch and the use of an icon's appearance to indicate the status of the program step represented by the icon while the program is being written.

It is thus an object of the invention to provide an improved computer system;

It is a further object of the invention to provide a novel visual programming system.

It is another object of the invention to provide a visual programming system which is easier to use than prior-art visual programming systems.

It is an additional object of the invention to provide a visual programming system in which the program is represented by means of a program structure which is interpreted to produce an iconic representation of the program structure and to execute the program.

It is a still further object of the invention to provide a visual programming system in which program structure and content are separately dealt with.

It is yet another object of the invention to provide a visual programming system in which the appearance of an icon representing a program step indicates the status of the program step during the program writing process.

It is a further additional object of the invention to provide4 a visual programming system in which the representation of the program is easily understood.

Other objects and advantages of the present invention will be understood by those of ordinary skill in the art after referring to the detailed description of a preferred embodiment contained herein and the drawings, wherein:

Brief Description of the Drawings

Fig. 1 is a block diagram of the hardware of presentation apparatus embodying the invention;

Fig. 2 illustrates the authoring screen of the invention;

Fig. 3 is a further illustration of the authoring screen showing the icon library;

Fig. 4 is a further illustration of the authoring screen showing pull down and pull right windows;

Fig. 5 is a further illustration of the authoring screen showing an icon structure;

Fig. 6 is a further illustration of the authoring screen showing a content editing window;

Fig. 7 is a further illustration of the authoring screen showing a further content editing window;

Fig. 8 is a further illustration of the authoring screen showing a complex icon;

Fig. 9 is a detailed illustration of a complex icon;

Fig. 10 is an illustration of an editing window for the compose operation;

Fig. 11 is a further illustration of the authoring screen showing an arrow icon;

Fig. 12 and 12A are detailed diagrams of a portion of a program structure of the present invention;

Fig. 13 is a detailed diagram of a node of the program structure;

Fig. 14 is a detailed diagram of the complex icon stack of the present invention;

Fig. 14A is a detailed diagram of control variables used in the present invention;

Fig. 15 is a diagram of the variable list used for program structure variables in the present invention;

Fig. 16 is a diagram of the library structure employed in the present invention;

Fig. 17 is a diagram showing the components of the present invention and their interaction; and

Fig. 18 is a diagram of the ASSIGN step node.

Reference numbers in the figures have three or more digits. The two least-significant digits are reference numbers within a drawing; the more significant are the drawing number. For example, the reference number 401 refers to item 1 first shown in drawing 4.

Detailed Description of a Preferred Embodiment

While the apparatus for icongraphically representing and executing a program of the present invention may be employed to represent and execute computer programs of any kind, it is employed to best advantage in situations where programming is typically done in high-level languages adapted to specific applications. These languages are generally designed so that experts in the application who are not expert programmers can write programs in their areas of expertise. One class of such languages is those used to create computer-based presentations, i.e., presentations which are given by executing a computer program. An important advantage of such presentations over slides, tapes, or films is tha they may be interactive, i.e., the form the presentation takes for a given audience may be governed by the input of the audience during the presentation.

Two areas in which computer-based presentations are important are marketing, where the interactive nature of the presentation permits the customer to choose the parts of the presentation which are of particular interest to him, and training, where the interactive nature of the presentation permits a student to tailor a training course to his own needs and rate of progress. In the preferred embodiment, the apparatus of the present invention is used to create and execute such computer-based presentations. The following discussion will first deal with the hardware used in the presentation apparatus of the preferred embodiment, then will show how the presentation apparatus appears to the user, and will finally disclose the internal structure of the apparatus.

1. Presentation Apparatus Hardware: Fig. 1

The hardware used in a preferred embodiment of the presentation apparatus is shown in Fig. 1. VDP 101 is a video disk player which includes an interface by which the video disk player may be controlled by a remote processor. In the preferred embodiment, VDP 101 may be a LDP 2000 disk player manufactured by Sony Corporation. CS 105 is a host computer system which stores the data structures and executes the programs by means of which the presentation apparatus is implemented. CS 105 in a preferred embodiment is one of the class of computers which executes the MSDOS operating system. Connected to CS 105 are a keyboard, KB 107, by means of which text and control codes may be input to CS 105, and moust 109, by means of which position data and control signals may be input to CS 105. In a preferred embodiment, the control signals are input by means of two buttons on mouse 109. MMGW 103 is a multi-media color graphics work station which can display color video and audio from VDP 101 and can display computer graphics, text, and a curso C 111 in response to data which it receives from CS 105. MMGW 103 can further overlay text and graphics onto the video display. MMGW 103 in a preferred embodiment is a KTX-1350N videotex workstation manufactured by Sony Corporation. The presentation apparatus may of course also be implemented using peripheral devices other than video disk players, other video disc players, other work stations, and other CSs and operating systems.

Flow of information between the components is shown by arraow in Fig. 1. Beginning with CS 105, CS 105 executes the presentation apparatus programs and receives inputs (DIN 119) for the programsas required from KB 107 and mouse 109. The programs produce work station control data (WSCTL) 117, video commands VCMD) 113, and SYNC signals 114. WSCTL 117 specifying curso position, MMGW 103 outputs C 111 on SCR 121; in response to WSCTL 117 specifying graphics or text, it outputs graphics or text on SCR 121. As shown by the arrowheads, WSCTL 117 further includes acknowledgements which MMGW 103 returns to CS 105.

VCMD 113 and SYNC 114 control operation of VDP 101. In response to VCMD 113, VDP 101 plays a video disk and outputs audio-video signals (AV) 115 to MMGW 103. MMGW 103 responds thereto by outputting video images on SCR 121 and audio via a speaker in MMGW 103. As previously mentioned, MMGW 103 may overlay C 111, text, and graphics onto audio and video from VDP 101. Sync signal 114 from CS 105 to VDP 101 synchronizes the audio and video with the graphics, and text.

2. User Interface of the Presentation Apparatus: Figs. 2-8

Users of the presentation apparatus are engaged in either authoring or executing a presentation. When a presentation is being executed, the user interface is that determined by the presentation itself, and is not of further interest here. When a presentation is being authored, the presentation apparatus presents the author with novel authoring interface termed herein the symbolic authoring interface. The interface has three main features:

1. The presentation is represented by means of a display of interconnected icons (i.e. graphic representations of objects in a computer system). The display is termed herein an icon structure. The icons in the icon structure represent steps in the presentation and their interconnections specify the sequence in which the steps are executed.
2. Each icon has a general type and may have a specific content. The types include author-defined types, and all icon types presently available are displayed in an icon library.
3. An icon structure is created or modified by means of an interactive icon structure editor. There are two kinds of editing: structure edition, in which icons are added to or deleted from the structure, and content editing, in which an icon is given contents or the contents are modified.

The symbolic authoring interface will be explained in detail using figures 2-8, which show typical authoring screens (ASCR) which appear on SCR 121 of MMGW 103 when an author is creating a presentation.

<u>3. The Authoring Screen: Fig. 2</u>

To use the presentation apparatus, the author first invokes the presentation apparatus program by means of a command input via KB 107 to CS 105. In response to the command, CS 105 begins executing the presentation apparatus program. After some initialization, the screen of Figure 2, ASCR(1)201, appears on SCR 121. Since ASCR(1) has many features shared by all screens of the symbolic authoring interface which show the icon structure, it will be explained in some detail.

First, somewhere in ASCR(1) 201, there will be found C 111, and C 111 may be moved around in ASCR(1) 201 by moving mouse 109. Authoring actions and alterations in the display are indicated in ASCR(1) by moving mouse 109 to a portion of ASCR(1) 201 which controls the action or display and then pushing one of the buttons on the mouse. This action is termed "clicking on an item" in the art. Generally, "clicking on an item" is done in the preferred embodiment using the right-hand button of mouse 109; however, for certain operations, the left-hand button is used. One portion of ASCR(1) 201 which contains items to be clicked on is menu bar 211, which shows the top level of authoring actions available. As will be explained in more detail later, an authoring action is specified by clicking on one of the words in menu bar 211. A pull-down window then appears which contains a list indicating what actions are available, and a further choice is made by clicking on an item in the list.

Another such protion is display bar 208. Display bar 208 controls the portion of the icon structure which is presently displayed, the size of the icons in the icon structure, and whether a representation of the icon library is presently being displayed. By clicking on a location to the left of icon size field 209, the author can decrease the size of the icons; by clicking on a location to the right, the author can increase the size. By clicking on one of the four fields in scroll bar 213, the author can scroll the authoring screen up, down, left, or right to see a different part of the icon structure. In ASCR(1) 201, the authoring screen is at the beginning (i.e., upper right hand corner) of the structure, and consequently, the only possible directions are down and right, as indicated by the arrows. By clicking on home 217, the author can return to the upper right hand corner of the structure, and by clicking on icon library field (ILIBF) 219, the author can cause the icon library to be displayed along the left-hand side the authoring screen. When the icon library is displayed, it can be made to disappear by again clicking on ILIBF 219.

Since the author has as yet neither written a presentation nor loaded a previously-written presentation, ASCR(1) 201 contains only a single icon. Icon 203 is the START icon. Like all icons 203, it has a label, ILABEL 207, giving the icon's name, and a picture, IPIC 205. The picture on the START icon is a house, indicating that it marks the home position to which one returns by clicking on home 207. In the preferred embodiment, the START icon is colored red. As will be explained in more detail later, the color of an icon in the icon structure indicates the icon's state. The color red indicates that the author has not currently selected the icon for editing and that the icon presently has no content.

4. Structure Editing: Figs. 3-5

To create an icon structure representing a presentation, the author adds icons 203 to the START icon. The source of the new icons 203 is the icon library, which is made visible by clicking on ILIBF 219. Figure 3 shows ASCR(2) 301 which results when an author clicks on ILIBF 219 in ASCR(1) 201. ASCR(2) 301 is the same as ASCR(1) 201, except that icon library (ILIB) 303 has appeared in a window at the left-hand side of the ASCR. ILIB 303 is a list of type icons (TICONs) 305. Each TICON 305 has the same components as ICON 203. The list is arranged in alphabetical order by ILABEL 207. Each TICON 305 represents a type of operation in the presentation. Only a portion of ILIB 303 is visible in the window; other portions of ILIB 303 are made visible by clicking on previous field (PRF) 307, which moves the window up the list, and next field (NF(309, which moves the window down the list. The TICONs 305 in ILIB 303 are red or green. The red icons are simple icons; as previously indicated, the red color indicates that they as yet have no user-defined content. In some cases, a red icon has no content at all; in others, it has a default content. User-defined content is added in a separate operation after the simple icon is added to the icon structure. The green icons are complex icons, which represent one or more simple or other complex icons; the simple icons in the complex icon may have user-defined content. The manner in which complex icons are created will be explained in detail latter.

The icon structure editing operations which are available to an author are select, which selects one or more icons 203 to be operated on, build, which adds an icon 203 of the type represented by a TICON 305 in ILIB 303 to the icon structure, cut, which removes an icon 203 from the structure and saves it for later use,,copy, which saves an icon 203 without removing it from the structure, and paste, which addes a saved icon 203 to the structure. The technique by which an author performs the operation is similar in each case, and will be explained in detail only for the build operation.

In a preferred embodiment, one icon 203 in the structure and one TICON 305 in ILIB 303 is always selected. The selected icon 203 and TICON 305 are distinguished by their appearance from other icons 203 and TICON 305; in a preferred embodiment, selected icons 203 and TICONs 305 are purple. If the author wishes to select another icon 203 or TICON 305 for his operation, he clicks on the selected icon. In the build operation, the selected TICON 305 indicates the type of icon 203 which is to be added to the structure and the selected icon 203 in the structure is the icon 203 preceding the point at which the icon 203 selected form ILIB 303 is to be added. In the build operation, the authors specifies the TICON 305 for the type of icon 203 to be added and the location in the icon structure at which it is to be added. The specification is by clicking on the TICON 305 and the icon 203 in the icon structure immediately preceding the point at which the new icon 203 is to be added.

Next, the author indicates that he wants to perform an editing operation. He does so by either clicking on the Edit field of menu bar 211 with the right-hand button or clicking anywhere with the left-hand button. As a result of either operation, ASCR(3) 401 shown in Fig. 4 appears. ASCR(3) 401 contains editing pull down window (EPDW) 403 appears at the position occupied by C 111. EPDW 403 contains a list of items indicating structure, content, and icon library editing operations. The author clicks on the item for the operation he wishes to perform, in this case the build operation. When he does so, the selected item changes from white letters on a black field to the reverse. Moreover, in the case of the bvuild operation, a structure editing pull right window (SEPRW) 405 appears. SEPRW 405 contains a list of items indicating further details about the operation.

In the case of the BUILD operation, the author must specify whether the new icon 203 is to be attached vertically (below) or horizontally (to the right of) to the icon 203 preceding the point at which the new icon 203 is to be added. Attachment must be vertical for all icons 203 which do not represent branches. If the icon 203 represents a branch, attachment may be either vertical, in which case the new icon is added below the branching icon, or horizontal, in which case it is added to the right of the branching icon. In the present example, attachment must be vertical, so the author clicks on "vertical" in SEPRW 405. The field then turns from dark to light. To execute the action, the author clicks on "OK" in SEPRW 405.

The result is ASCR(4) 501 of Fig. 5. Windows 403 and 405 have disappeared and the new icon 203, in this case, a graphic icon 203 specifying a graphics display based on a graphics file provided by CS 105 to MMGW 103, appears below the START icon 203. The two icons 203 make up a simple icon structure (ISTRUC) 505. The icons in the structure are connected by link line 503 which specifies the order in which the program steps represented by the icons 203 Of ISTRUC 505 are to be executed. In the preferred embodiment, link lines 503 may be either vertical or horizontal. A vertical link line indicates that the icon 203 at the lower end of link line 503 will be executed after the icon 203 at the upper end. A horizontal link line indicates a branch. When the branch is taken, the icon 203 at the right end of the horizontal like line is executed next. ISTRUC 505 may be further edited in the manner described above: icons 203 may be added from ILIB 303 by means of the build operation, or may be copied, moved, or deleted within ISTRUC 505 by means of the copy, cut, and paste operations. Using the "range" field of EPDW 403, the author may select a sequence of icons 203 in INSTRUC 505 to be copied or cut. All of the icons 203 in the sequence turn purple.

A number of fields in EPDW 403 and SEPRW 405 are common to most of the menus in the presentation apparatus. The author clicks on the "OK" field when he has finished setting up for an operation and wishes the operation to be performed. By clicking on the "cancel" field, the author can cancel as much of the operation as he has already set up and start over. To find out more about the operation he is attempting to perform, the author clicks on the "help" field; when he does so, a help window appears with more information about the operation controlled by the window the author was working on when he clicked on "help".

ISTRUC 505 of Fig. 5 consists only of a vertical sequence of icons 203; however, ISTRUC 505 may generally include vertical sequences of icons 203 and branches to the right from two kinds of icons 203, complex icons and conditional icons. The branching structure of ISTRUC 505 is shown in Fig. 9. CXSTRUC 901 of that figure is a complex branch dependent from a complex MENU icon (CXICON) 803. It includes another complex branch (CXBR) 909 dependent from an IF(N) CXICON 803, and CXBR 909 includes a conditional branch (CONDBR) 907 dependent from an IF icon simple icon (SICON) 902. As may be seen from Fig. 9, branch may itself have branches.

The kinds of branches differ in the manner in which they are executed. A CXBR 909 is always taken. Its execution is like that of a procedure: after the branch has been executed, there is a return to the icon following the complex icon from which the branch is dependent and execution continues at that point. Conditional branches, initiated by IF SICONs 902, are taken only when a condition specified in the IF icon's content is false. When the conditional branch is contained in a complex icon branch, execution returns from the complex icon branch when it reaches the end of the conditional branch; if the conditional branch is not contained in a complex icon branch, the presentation is terminated when the end of the conditional branch is reached.

5. Content Editing: Figs. 6 and 7

When an icon 203 is added to ISTRUC 505, it has a default content. The default content is sufficient to permit exeuction of the presentation corresponding to ISTRUC 505. An author may further define the icon 203's content by selecting the icon 203 (ranges may also be selected), then calling up EPDW 403 as previously described, and then clicking on the "content" field. Thereupon, a content-editing pull right window appears which has the fields "all", "non-filled", "help", "cancel", and "ok". The three latter fields are analgous to those already described for SEPRW 405. The author clicks on "all" and "ok" when he wishes to be able to edit all author-definable aspects of icon 203's content and on "non-filled" and "ok" when he wishes to edit those aspects which have not yet been defined. In either case, a content editor window 601 appears on the authoring screen which contains a menu indicating content editing actions specific to the type of icon 203 which was activated.

Assuming that ISTRUC 505 shown in Fig. 5 is being edited, the icon 203 to which content may be added is the GRAPHIC icon 203. That icon 203 represents the display of a graphics file on MMGW 103. The source of the graphics file is CS 105. Figure 6 shows content editor window 601 for the GRAPHIC icon 203. The fields in the window are the following:

Status field 603 indicates whether the GRAPHIC icon 203 being defined already has content. If it does not, field 603 has the value "new".

Define field 605: when the author clicks on the field and one of the other fields, he indicates that he wishes to select from the choices available to fill the other field.

File Name field 607: When icon 203 is defined, this field contains the name of the file in CS 105 which contains the graphic to be displayed.

OK 609, HELP 611, and CANCEL 613: These fields have functions analogous to the fields of the same names in SEPRW 405.

In the case of a GRAPHIC icon 203, the icon 203's content is the name of a file containing the graphic. Defining the content thus involves only one field, file name field 607. If a file already exists and the author knows the file's name, the author simply clicks on "file name" field 607 and then uses KB 107 to input the file name. When he is finished, he clicks on the "ok" field. Window 603 then disappears and ASCR(4) 501 reappears, except that GRAPHIC icon 203 has changed color from red to yellow, indicating that it now has content. Additionally, a character string may appear on the icon which indicates the kind of content.

If there is as yet no file or if the author does not know the file's name, the author clicks on "define" field 605 and then on "file name" field 607. When he does so, define window 701 shown in Fig. 7 appears. Window 701 is titled "Choices" and offers three fields from which the author may choose: "file names" 703, "variables" 705, and "graphiz ed" 709. If the author clicks on "file names 709", a window showing a list of file names appears, and the author can select a graphics file from the list; if he clicks on "variables" 705, a list of system and user-defined variables appears and the author selects one of the variables. An ASSIGN icon 203 permits a file name to be assigned to a variable, and the file name assigned to the selected variable will be the name of the graphics file for the graphics icon. The use of variables to specify graphics files permits the presentation to determine on the basis or interactions with the user of the presentation which graphics the user will see at a later point. If the author clicks on "graphix ed" field 709, the presentation apparatus invokes a graphics editor by means of which the author can create or edit a graphics file. The created or edited file then becomes the content of GRAPHICS icon 203. Again, once icon 203 has content, it changes color from red to yellow.

While content is added to a given icon 203 as required by the icon's type, the sequence just explained for the GRPAHIC icon 203 is typical of content editing for all icon types. Three aspects are particularly noteworthy: first, the author can always see the choices which are presently available to him when filling in a field. Second, the editing is type sensitive: the windows which appear during the editing process depend on the type of icon 203 being edited and the facilities made available to the author during content editing are only those required for the type of icon 203 which he is editing. For example, there is a PAUSE icon 203 which specifies a pause in the presentation; the content for PAUSE is a value representing seconds of time; the windows give the author the choice of inputting a constant representing the number of seconds and specifying a variable whose value is the number of seconds. Third, the choices available to an author generally include an interactive editor which permits him to directly specify the content. For example, there is a PLAY icon which specifies that a disk in VDP 101 be played up to a certain frame; the content editor for PLAY permits the author to interactively play the video disk and mark the frame at which playing is to end. The content editor then derives the frame number from the mark.

6. Complex Icons: Figs. 8-10

A complex icon is an icon which represents a sequence of one or more others icons, including other complex icons. When a complex icon is selected from ILIB 303 and added to ISTRUC 505, the sequence of icons represented by the complex icon become a branch of ISTRUC 505. The branch is always taken, and when execution of the branch is complete, execution of ISTRUC 505 resumes at the icon following the complex icon.

The presentation apparatus of the present invention includes system and user-defined complex icons. Fig. 8 shows one such complex icon, the system-defined MENU complex icon (CXICON) 803 in ILIB 303. The author can identify a CXICON 803 in ILIB 303 by its color: complex icons are green and simple icons are red. The action defined by the MENU CXICON 803 is the following: first the presentation apparatus outputs a graphic containing a menu which lists possibilities. Then, the presentation apparatus receives input indicating which of the possibilities the user has chosen, and finally, the presentation branches in response to the selected input.

An author adds a CXICON 803 to ISTRUC 505 the same way he adds a simple icon 203: by clicking on CXICON 803 and the icon preceding the point in ISTRUC 505 at which he wishes to add CXICON 803, calling up EPDW 403, clicking on "build", clicking on the proper field of SEPRW 405, and clicking on "ok" in SEPRW 405. When the author has performed these actions, the structure of icons shown in Fig. 9 is added as a branch from the specified point in ISTRUC 505. As a complex icon branch, the structure is executed completely and execution then continues at the icon 203 following CXICON 803.

The structure associated with the MENU CXICON 803 appears as complex struction (CXSTRUC) 901 in Fig. 9. CXSTRUC 901 for the MENU CXICON 803 consists of MENU CXSTRUC 903, which is made up of the MENU CXICON 803, the loop start (L-START), GRAPHIC, INPUT, and L-END simple icons (SICON) 902, and the IF(N) CXICON 803, representing a multi-way branch. the IF(N) CXICON 803 in turn consists of IF(N) CXSTRUC 905, which is made up of four IF or conditional branching SICONs 902, an ERROR exit SICON 902, and an EXIT SICON 902. As may be seen from Fig. 9, CXSTRUC 905 is nested in XCSTRUC 903. Such nesting may occur to any depth. CXICONs 803 in ISTRUC 505 in a preferred embodiment serve only to label the branch represented by the CXICON 803 and mark the branch's beginning; the actual steps of the presentation are represented by the branch's SICONs 902. Thus, in CXSTRUC 901, L-START and L-END define the beginning and ending of a loop in the presentation. On each execution of the loop, a graphic of the menu is displayed (the GRAPHIC SICON 902), an input is received from the user (the INPUT SICON 902), and the input is tested in the IF SICONs 902. If the condition of one of the IFs is satisfied, the branch beginning at that IF SICON 902 is executed. If the branch includes an EXIT SICON 902, the loop is exited and execution continues at the icon 203 following the MENU CXICON 803 in ISTRUC 505; otherwise, it is repeated. If the condition of none of the IF SICONs 902 is satisfied on an execution of the loop, the action specified for the ERROR SICON 902 is executed and execution continues at the icon 203 following the MENU CXICON 803.

Structure and content editing for a CXICON 803 may involve either the entire CXSTRUC 901 represented by CXICON 803 or its component icons 203. A range cannot be specified for an editing operation which involves both icons 902 contained in CXSTRUC 901 and icons 902 outside of CXSTRUC 901. For example, if CXICON 803 is specified in a cut operation, the entire CXSTRUC 901, including any nested CXSTRUCs, represented by CXICON 803 will be removed; additionally, a cut operation may remove icons 203 within CXICON 203; however, a range may not be specified for a cut operation which includes some icons 203 within CXSTRUC 901 and others outside of CXSTRUC 901. Similarly, content editing may be done on CXICON 803, in which case the content editor provides the icons 203 making up CXSTRUC 901 to the author one by one for content editing, or it may be done on individual icons 203 making up CXSTRUC 901.

To add a CXICON 803 to ILIB 303, the author first defines CXSTRUC 901 for the new CXICON 803 by using the mouse and the "range" field of EPDW 403 to define a range of icons 203 in ISTRUC 505 as previously described. Then, the author clicks on the "compose" field of EPDW 403, and the window shown in Fig. 10 appears. COMPOSE EDITOR window 1001 includes a field 1003 for defining ILABEL 207 for the icon, a field 1005 for defining IPIC 205, and a field 1007 for a short comment about the new CXICON 803. To define each of these fields, the author clicks on it. In the case of field 1003, he enters the character string for the new CXICON 803's ILABEL into the field. In the case of field 1005, he specifies a letter of the alphabet. The presentation system has associated a graphic symbol with each letter of the alphabet and uses the graphic symbol associated with the specified letter in the new CXICON 803. The input for remarks field 1007 is a character-string comment about the icon. When the fields are filled in, the author clicks on the OK field, and the new CXICON 803 is added to ILIB 303.

When a new CXICON 803 is defined, its component SICONs 902 retain whatever content they had in the ISTRUC 505 in which the were defined. It is thus possible to define CXICONs 803 with varying amounts of content, ranging from CXICONs 803 in which none of the SICONs 902 has had content defined for it through CXICONs 803 in which the content of all of the SICONs 902 is completely defined. This property of CXICONs 803 may be used to create templates for defining members of families of complex operations where all members share certain parts of the content but differ from each other with regard to other parts of the content. By clicking on the "non-filled" field in the pull right window which appears when content editing is specified, the author can indicate that he wishes only to define the undefined parts of the content of a CXICON 803 which has been newly-added to ISTRUC 505.

To remove a CXICON 803 from ILIB 303, the author clicks on the CXICON 803 to activate it and then calls up EPDW 403 and clicks on the "decompose" field. When the field turns from black to white, he clicks on the "ok" field. In response thereto, the activated CXICON 803 vanishes from ILIB 303. Removal of a CXICON 803 from ILIB 303 does not, however, affect instances of the CXICON 803 in ISTRUC 505. Similarly, redefining a CXICON 803 by removing it from ILIB 303 and then adding the same CXICON 803 with a different definition does not change the instances of the CXICON 803 in ISTRUC 505.

7. Condensing the Display of ISTRUC 505: Fig. 11

In a preferred embodiment, two techniques are used to condense the display of ISTRUC 505 on SCR 121. First, an author may reduce the size of the icons 203 displayed on SCR 121 and thereby increase the portion of ISTRUC 505 which will fit on a single screen. To change the size of the icons, the author clicks on icon size field 209 with the left button of mouse 109. When the size of the icons has been reduced below a certain point, IPIC 205 for the icon is not displayed; when the size is reduced still further, ILABEL 207 is not displayed. In other embodiments, the author may reduce and increase the size of TICONs 305 in ILIB 303 in the same fashion as that just described.

The presentation apparatus further automatically provides the ARROW icon shown in Fig. 11. An ARROW icon 1103 indicates that there are further icons 203 in ISTRUC 505 which are off of SCR 121 in the direction pointed by the arrow. IPIC 205 in ARROW icon 1103 contains a down or right arrow and ILABEL 207 contains the name of the first icon 203 which is off the screen in the direction pointed by the arrow. In Fig. 11, ILABEL 207 specifies the TV CXICON icon 803. In order to see the icons 203 pointed to by ARROW icon 1103, the author need only click on ARROW icon 1103. In a preferred embodiment, ARROW icons 1103 are colored blue to differentiate them from icons 203 which are simply the last icon 203 in a branch. The differentiation by color is particularly useful when the icons' size has been reduced so that ILABLE 207 and IPIC 205 are not readable.

In the presently-preferred embodiment, all icons making up a CXICON 803 are displayed in ISTRUC 505; however, in other embodiments, the author may be able to determine a level at which the contents of a CXICON 803 will not be displayed. For example, referring to Fig. 9, if a MENU CXICON 803 were at that level, all that would appear of the CSICON 803 would be the MENU icon itself. If the same MENU

CXICON 803 were one level above that level, the SICONs 901 in IF(N) CXSTRUC 905 would not appear.

8. Running, Saving, and Loading a Presentation

While an author is working on a presentation, he may run all or part of it by selecting a range in ISTRUC 505 indicating the portion to be run, clicking on the "control" field of menu bar 211, and clicking on the "run" and "ok" fields of the pull-down menu which appears when he clicks on "control". The presentation apparatus then executes the steps of the presentation specified by the selected portion of ISTRUC 505. As previously mentioned, the default content of an icon 203 guarantees that the step represented by it will be executable even though no further user-defined content has been added. Additionally, presentations may be run on versions of the presentation apparatus which do not permit editing of the presentation.

When an author is finished working on a presentation, he can save ISTRUC 505 in a MSDOS file by clicking on the "file" field of menu bar 211 and clicking on fields of the pull down menu which results to save the course. If he wishes to discard what he has done during an editing session, he clicks on an "abort" field which discards the changes. By means of the same pull down menu, the author can load a presentation which he has previously saved. After an author has saved or aborted the result of an editing session, he can exit the presentation apparatus by clicking on the "maestro" field and then clicking on the "exit" and "ok" fields of the pull down menu which results.

9. Overview of the Program Structure of the Presentation Apparatus: Figs. 12 and 12A

In a preferred embodiment of the presentation apparatus, ISTRUC 505 and the execution of the presentation are both the result of the interpretation of a single data structure, the program structure. Moreover, when structure and content editing operations are performed, the operations are in fact performed on the program structure. A portion of a program structure is shown in overview in figures 12 and 12A.

What is shown in those figures is the portion of program structure (PSTRUC) 901 which implements a MENU CXICON 803. PSTRUC 1201 is made up of nodes 1203 linked together by pointers. There are two functional types of nodes: icon nodes (INODEs) 1205, representing icons 203, and content nodes (CNODEs) 1207 representing the contents of an INODE 1205. In Fig. 12, INODEs 1205 are represented by square or diamond boxes and CNODEs 1207 are represented by oblong boxes. There are two types of INODEs 1205, corresponding to the two types of icons 203: SINODEs 1208 corresponds to SICONs 902 and CXNODEs 1207 correspond to CXICONs 803.

INODEs 1205 are linked to each other by pointers which appear in Fig. 12 as structure links (SLINKS) 109, shown as heavy lines. A special type of SLINK 1209 is CSLINK 1221, which points to the first INODE 1205 dependent from CXNODE 1207. Conceptually, a given INODE 1205 may have one or more of three different types of SLINKS 1209: a DOWN SLINK 1209, a RIGHT SLINK 1209, and a CSLINK 1221. If a given type of SLINK 1209 is not needed in a given INODE 1205, its place is taken by a NULL SLINK 1209.

A SINODE 1208 may have one of more CNODEs 1207 dependent from it. The CNODEs 1207 dependent from a given SINODE 1208 form a binary tree. The CNODEs 1207 are linked to each other and to the SINODE 1208 by pointers which appear in Fig. 12 as content links CLINKs) 1211, shown as light lines. Each INODE 1205 additionally has a procedure pointer (PPTR) 1210, shown as an arrow with a right angle, which points to the executable code for a procedure. When SINODE 1208, executes the operation, and returns SLINK 1209 to the next INODE 1205 in PSTRUC 1201 to be executed.

As can be seen from Fig. 12, PSTRUC 1201 is a branched tree of INODEs 1205. There are two kinds of branches, corresponding to the two kinds of branches in ISTRUC 505. Complex branch (CXBR) 1223 is connected by CSLINK 1209 to a CXNODE 1207; conditional branch (CONDBR) 1225 is connected by a RIGHT SLINK 1209 to an IF INODE 1205 and represents the branch taken when a test in IF INODE 1205 tests true. Additionally, those of the INODEs 1205 which are SINODEs 1208 may have dependent binary trees of CNODEs 1207.

As will be explained in more detail later, the presentation apparatus includes a number of programs which walk PSTRUC 1201 and use the information contained therein to display ISTRUC 505 corresponding to PSTRUC 1201, to modify the structure or content of PSTRUC 1201, to execute the presentation represented by PSTRUC 1201 recursively walks the CXBRs 1223. When the interpreter encounters a CXNODE 1207, it executes the procedure pointed to by PPTR 1210, which in the case of CXNODE 1207, saves SLINK 1209 to the INODE 1205 following the CXNODE 1207 in a stack and reinvokes the interpreter using CSLINK 1221 to the first INODE 1205 in CXBR 1223 beginning at CXNODE 1207. The interpreter then deals with the INODEs 1205 in the branch. When it executes a SINODE 1208 in the branch, it invokes the procedure pointed to by PPTR 1210, which recursively walks the dependent binary tree of CNODEs 1207 to obtain the data stored therein and uses the data to carry out the program step specified by the SINODE 1208. When the interpreter reaches the end of the branch, indicated by a null SLINK 1209, it returns and continues execution with the saved SLINK 1209. If the CXBR 1223 contains a CXNODE 1207, the interpreter again recurses as described above.

Thus, in PSTRUC 1201 of Fig. 12, the interpreter saves DOWN SLINK 1209 of the MENU CXNODE 1207 and recurses using CSLINK 1221, which points to another CXNODE, LOOP CXNODE 1207. The interpreter again recurses, saving LOOP CXNODE 1207's null DOWN SLINK 1209 as it does so. This time, CSLINK 1221 leads to L-START SINODE 1208. Execution then continues following the DOWN SLINKs 1209 until IF(N) CXNODE 1207 is reached, at which point the interpreter again recurses, saving DOWN SLINK 1209 of IF(N) CXNODE 1207. The interpreter then follows one of the CONDBRs 1225, as determined by the tests performed in the IF SINODEs 1208. At the end of the branch, marked by a NULL DOWN SLINK 1209, the interpreter returns and follows the DOWN SLINK 1209 of IF(N) CXNODE 1207. When execution of the loop terminates, the interpreter returns to LOOP CXNODE 1207, and since LOOP CXNODE 1207 has a null DOWN SLINK 1209, to MENU CXNODE 1207.

In executing each SINODE 1208, the interpreter follows the SINODE 1208's and uses the contents of the CNODEs 1207 in the execution of the procedure pointed to by PPTR 1210. The execution of GRAPHIC SINODE 1208 may serve as an example here. GRAPHIC SINODE 1208 has two CNODEs 1203 dependent from it. One, labelled GRAPHICS FILE, contains a value which resolves to the name of a graphics file. When the interpreter invokes the procedure pointed to by PPTR 1210, the procedure uses the name of the graphics file and the type of the graphics file to invoke the proper program for providing the contents of the graphics file to MMGW 103 for display.

10. Detailed Structure of Node 1203: Fig. 13

All of the nodes 1203 employed in PSTRUC 1201 in a preferred embodiment have the detailed structure shown in Fig. 13. The description of the fields of node 1203 begins the SFLAGS 1301. SFLAGS 1301 contains flags which indicate the current state of node 1203. Included are the following:

an IN RANGE flag, indicating whether the node 1203 is within a range of nodes defined in ISTRUC 505;

an active flag, indicating whether the node 1203 marks the point at which a BUILD or PASTE operation is to occur;

a CONTENT flag, indicating whether a content editing operation has yet been performed on node 1203;

a MARK flag, indicating that node 1203 is to be copied; and

a PARSE flag, indicating whether node 1203 has in fact been copied in the current copy operation.

When an icon 203 corresponding to an INODE 1205 with a SELECTED or INRANGE flag set is displayed, icon 203 is purple; similarly, when the CONTENT flag is set, but the SELECTED or INRANGE flag is not set, icon 203 is green; otherwise, icon 203 is red.

In an INODE 1205, the value of the next field, ITYPE 1301, indicates the type of the INODE 1205. It further specifies the picture that is to be displayed in IPIC 205 of icon 203 corresponding to INODE 1205. The next two fields, NDATA 1309, are used to hold the contents of the node. In INODEs 1205, INAME field 1305 contains the character string used in ILABEL 207 when the icon 203 corresponding to the INODE 1205 is displayed. IBUFFER 1307 is used in INPUT INODEs 1205 to hold a prompt for the desired input; in CNODEs 1207, IBUFFER 1307 holds data to be used by the procedure pointed to by PPTR 1210. For example, in the CNODEs 1203 dependent from the GRAPHIC SICON, IBUFFER 1307 in one CNODE 1203 contains a file name or the name of a variable representing a file name; in the other, it contains the file type.

When an icon 203 corresponding to an INODE 1205 with a SELECTED or INRANGE flag set is displayed, icon 203 is purple; similarly, when the CONTENT flag is set, but the SELECTED or INRANGE flag is not set, icon 203 is green; otherwise, icon 203 is red.

In an INODE 1205, the value of the next field, ITYPE 1303, indicates the type of the INODE 1205. It further specifies the picture that is to be displayed in IPIC 205 of icon 203 corresponding to INODE 1205. The next two fields, NDATA 1309, are used to hold the contents of the node. In INODEs 1205, INAME field 1305 contains the character string used in ILABEL 207 when the icon 203 corresponding to the INODE 1205 is displayed. IBUFFER 1307 is used in INPUT INODEs 1205 to hold a prompt for the desired input; in CNODEs 1207, IBUFFER 1307 holds data to be used by the procedure pointed to by PPTR 1210. For example, in the CNODEs 1203 dependent from the GRAPHIC SICON, IBUFFER 1307 in one CNODE 1203 contains a file name or the name of a variable representing a file name; in the other, it contains the file type.

The next fields contain three sets of pointers corresponding to SLINKs 1209, CLINKs 1211, and PPTRs 1210. If any of the pointers is not being used in node 1203, the pointer's field contains a null value. Beginning with PPTRs 1210, all INODEs 1203 have a pointer in PROC PTR 1311 which points to the procedure which is invoked when INODE 1203 is executed. INODES 1203 which perform computations also have a pointer in EPROC PTR field 1313 which points to a function which performs the computation and returns the result. In the presentation apparatus of the preferred embodiment, there are two such INODEs 1203: the IF INODE and the ASSIGN INODE. In the IF INODE 1203, the function computes the value which the ASSIGN INODE 1203 assigns to a variable. Continuing with CLINKs 1211, each node 1203 has two pointers which may point to CNODEs 1207. As previously explained, the data in the CNODEs 1207 is used in the procedure pointed to by PROC PTR 1311, and consequently, the fields are termed ARG1 PTR 1317 and ARG2 PTR 1319.

INODES 1205 use 5 fields for SLINK pointers 1209. The pointers include DOWN PTR 1323, which is the DOWN SLINK 1209, RT PTR 1327, which is the RIGHT SLINK 1209, the CSPTR 1329, which is CSLINK 1221. Additionally, an INODE 1205 may have an UP PTR 1321, which points to the INODE 1203 above the INODE 1203 in question, and a LEFT PTR 1325, which points to the INODE 1203 to the left of the INODE 1203 in question. The UP and LEFT pointers permit an author to move both forward and backward in PSTRUC 1201 when he is editing a presentation.

The final two fields are the following: RESPTR 1331 is a pointer which is reserved for future use. DISPD 1333 contains data used in displaying ISTRUC 505 corresponding to PSTRUC 1201. The field contains three sub fields, as shown in the detail in Fig. 12:

> ARROWFL 1335 is a field indicating whether the icon 203 corresponding to INODE 1205 should be displayed as a down or right arrow icon.
>
> YCOORD 1337 is a field indicating the position of the icon 203 corresponding to INODE 1205 on the vertical or Y axis of a coordinate system defining positions of icons 203 in "icon space".
>
> XCOORD 1339 is a field indicating the

position of icon 203 corresponding to INODE 1205 on the horizontal or X axis of the coordinate system.

When ISTRUC 505 is displayed, a portion of the icon space defined by the coordinate system is displayed on SCR 121 of MMGW 103. If the icon 203 corresponding to INODE 1205 falls on the left or bottom edge of the portion being displayed, the information in ARROWFL 1335 is used to determine which arrow icon 1103 is to be displayed.

11. Other Data Structures used in the Presentation Apparatus: Figs. 14-16

In addition to PSTRUC 1201, a preferred embodiment of the presentation apparatus employs three other data structures and a set of variables which are important for an understanding of the invention. The first of the data structures, complex icon stack 1401, shown in Fig. 14, is the stack employed by the recursive interpreters while walking PSTRUC 1201. In a preferred embodiment, complex icon stack 1401 is implemented as a linked list of complex icon stack frames, (CXSTKFs) 1403. Each CXSTKF 1403 has three fields: CXPTR 1405, which is a pointer to CXNODE 1207 being interpreted in the recursion represented by the CXSTKF 1403, CRSPTR 1407, which is DOWN PTR 1323 from CXNODE 1207 being interpreted in the recursion, and PRPTR 1409, which is a pointer to the CSXTKF 1304 for the previous recursion. Complex icon stack 1401 may of course alse be implemented using other stack implementation techniques such as a hardware stack.

The set of variables is shown in Fig. 14A. For convenience, they are shown as belonging to a single data structure, but they need not be so organized. The variables are the following:

STATE FLAGS variable 1413 includes flags indicating whether the presentation is being edited or run, whether content editing or structure editing is taking place, and whether IL 303 is being displayed.

LOOP FLAGS 1415 includes flags indicating that a loop is being executed and that an EXIT INODE 1205 specifying a loop exit has been executed.

SCR WIDTH 1417 and SCR HEIGHT 1419 indicate the size of SCR 121.

SCRX 1421 and SCRY 1423 indicate the position of SCR 121 in the "icon space".

ICONIX 1425 and ICONY 1427 indicate the coordinates of the current INODE 1205 in "icon space".

ISIZE 1429 indicates the size of the icons in ISTRUC 505.

ILSIZE 1430 indicates the size of icons 203 in ILIB 303 in embodiments where that is adjustable.

HEAD_PTR 1431 is the pointer to the first node 1203 in PSTRUC 1201.

CURR_PTR 1433 indicates the node 1203 interpreted during an execution of the presentation represented by ISTRUC 505.

REF_1435 is a pointer to the INODE 1205 from which the operation of displaying ISTRUC 505 commences.

S_RANGE_PTR 1437 and E_RANGE_PTR 1439 are pointers to the first and last INODEs 1205 in a selected range.

HOLD_PTR 1443 and LOOP_PTR 1445 are pointers to positions in a loop.

LIB B PTR 1446 is a pointer to a structure in the icon library representing an icon.

LIB H PTR 1447 is a pointer to the beginning of ILIB 303.

LIB REF PTR 2449 is a pointer identifying the first icon 203 to be displayed in ILIB 303.

The second of the data structures is variable list 1501, shown in Fig. 15. As previously mentioned, values may be represented in CNODEs 1207 by variable names which are global to the nodes of PSTRUC 1201; in other embodiments, variable names may also be local to a CXBR 1223. A variable name may represent either a single value or a compound value such as an array or structure. In the preferred embodiment, there are two kinds of variables: built-in system variables and user-defined variables. A user may define a variable whenever he performs a content editing operation on a SINODE 1208 which assigns a value to a variable. In a preferred embodiment, the SINODEs 1208 which perform this function are the ASSIGN SINODE 1208, which explicitly assigns the value of an expression to a variable, and the INPUT SINODE 1208, which assigns input received from KB 107 or mouse 109 to a variable.

The first time a SINODE 1208 which creates a variable is executed, an entry for the variable is created in variable list 1501. Variable list 1501 is a linked list of variable list entries (VLE) 1503. In the case of variables representing a single value, there is one VLE 1503; in the case of variables representing compound values, there is a VLE 1503 for the variable name in variable list 1501 and branching therefrom a chain of VLEs 1503 containing the variable's values. The fields in VLE 1503 are the following:

VNAME 1505: The character-string name of the variable. In a preferred embodiment, variable names begin with the @ character. In VLEs 1503 representing elements of a compound variable, VNAME 1505 contains the element name. for example, in elements of arrays, VNAME 1505 contains the element's index.

VTYPE 1507: the data type of the value represented by the variable or element.

VVAL 1509: the value of the variable or element.

NPTR 1511: In VLEs 1503 representing an entire variable, a pointer to the next VLE 1503 in variable list 1501; in VLEs 1503 representing an element, a pointer to a chain of VLEs 1503 representing elements dependent from that element..

RPTR 1512: if the variable is compound, RPTR 1512 points to VLE 1503 for the next element at the same level as the given element.

In the preferred embodiment, a hash function (HASHF) 1515 and hash table (HT) 1521 are used to obtain rapid access to a VLE 1503 for a variable. When a VLE 1503 for a variable is added to variable list 1501, variable name 1513 for the variable is input to HASHF 1515. HASHF 1515 computes a hash table index (HTI) 1517 from the variable name, which it uses to locate hash table entry (HTE) 1523 in HTI 1517. If variable name 1513 hashes to the same HTI 1517 as a variable name 1513 for which there is already a VLE 1503 in VL 1501, HTE 1523 for that HTI 1517 will contain VLEPTR 1519 to VLE 1503 for which there is a HTE 1523. When a variable name is encountered in a CNODE 1207 during execution of PSTRUC 1201, the variable name is input to HASHF 1515, the resulting HTI 1517 used to locate HTE 1523, and the VLEPTR 1519 therein used to locate a VLE 1503 for a variable name 1513 which hashes to the same HTI 1517 as the given variable name 1513, VL 1501 is searched beginning at VLE 1503 until VLE 1503 for the desired variable name is located.

The third data structure is library structure 1601. Library structure 1601 is a variation on PSTRUC 1201 which is used to represent ILIB 303. Each TICON 305 in ILIB 303 is represented by a socket CXNODE 1605 whose DOWN PTR 1323 points to the next CXNODE 1605 in library list (LIBLIST) 1603 making up ILIB 303 and whose UP PTR 1321 points to the previous CXNODE 1604 in LIBLIST 1603. CXPTR 1329 in socket CXNODE points to the first node 1203 in a structure of nodes 1203 representing the TICON 305 being defined. If TICON 305 is a SICON 902, the structure of nodes 1203 is simply SINODE 1208 representing the SICON 902; since a TICON 305 for a SICON 902 has no content, SINODE 1208 will have no CNODES 1207 dependent from it. If TICON 305 is a CXICON 803, the structure of nodes is a copy of the portion of PSTRUC 1201 specified by the author when he defined CXICON 803. That copy appears as CXPSTRUC 1607 in Fig. 16. Any CNODEs 1207 belingong to the copied portion of PSTRUC 1201 will also be included in CXPSTRUC 1607.

12. Overview of Presentation Apparatus Components: Fig. 17

Figure 17 presents an overview of the components of the presentation apparatus of the present invention and of the relationships between them. In Fig. 17, rectangular blocks represent component data structures, circles represent functionally-grouped component programs, and arrows represent data flows between data structures and programs and between programs. Data structures and programs are present to the extent required in the memory of CS 105 when a presentation is being edited or executed.

Discussion of the presentation apparatus will begin with the data structures. LSTRUC 1601, implementing the icon library, PSTRUC 1201, and VARLIST 1501 have already been discussed in detail. control data (CDATA 1721) includes CXSTACK 1401 and control variables (CVs) 1411, which also have been discussed in detail. Paste buffer (PBUFF) 1723, finally, is an area of the memory of CS 105 which is used for temporary storage of structures being copied between different points in PSTRUC 1201.

The programs have the following functions:

SED 1702 is the structure editor. It edits the structural aspects of PSTRUC 1201 and LSTRUC 1601.

CED 1703 is the content editor It edits the content of PSTRUC 1201.

SDISP 1705 interprets PSTRUC 1201 and LSTRUC 1601 to provide the displays of ISTRUC 505 and ILIB 1705 corresponding to PSTRUC 1201 and LSTRUC 1601.

RUN 1709 interprets PSTRUC 1201 to execute the presentation represented by PSTRUC 1201.

DEV 1701 receives inputs from SED 1702, CED 1703, SDISP 1705, and RUN 1709 as required bo produce WSCTL 117, SYNC 114, and VCMD 113 controlling workstation MMGW 103 and video disk playber VDB 101. DEV CTL 1701 also receives inputs DIN 119 from KB 107 and MOUSE 109, which it provides to SDISP 1705, SED 1702, CED 1703, and RUN 1709. In other embodiments, DEV CTL 1701 may control other devices attached to CS105.

SAVE/LOAD 1711 interprets PSTRUC 1602, LSTRUC 1201, and VARLIST 1501 as required to save the structures and the list of variables on files in a magnetic disk belonging to CS105 and also interprets such files to load LSTRUC 1601, PSTRUC 1201, and VARLIST 1501 from the disk to the memory of CS 105. As indicated by the arrow from RUN 1709, SINODEs 1208 in PSTRUC 1201 may specify that SAVE/LOAD load and save PSTRUCs 1201 and VARLISTs 1501 during execution of a presentation.

The central role of PSTRUC 1201 in the presentation apparatus is clear from the data flows indicated in Fig. 17. SED 1702 reads PSTRUC 1201 n; and alters SLINKs 1209 therein; CED 1703 reads

PSTRUC 1201 and adds and modifies CNODEs 1207 and CLINKs 1211. SDISP 1705 interprets PSTRUC 1201 to produce ISTRUC 505. RUN 1709, interprets PSTRUC 1201 to produce the presentation itself, and SAVE/LOAD 1711 interprets PSTRUC 1201 to produce the file in which a representation of PSTRUC 1201 is saved. The role of LSTRUC 1601 is also apparent from the arrows: SED 1702 defines complex icons by copying portions of PSTRUC 1201 to LSTRUC 1601, where they become CXPSTRUCs 1607 to PSTRUC 1201. SDISP 1705 interprets LSTRUC 1601 to produce ILIB 305.

In editing the content of PSTRUC 1201, CED 1703 also defines variables in VARLIST 1501; when RUN 1709 interprets PSTRUC 1201, it sets variables in VARLIST 1501 as specified in PSTRUC 1201 and executes PSTRUC 1201 as determined by the values of the variables. As previously mentioned, interpretation of complex icon branches in PSTRUC 1201 and LSTRUC 1601 may be recursive; consequently, SED 1702, CED 1703, RUN 1709, and SAVE/LOAD 1711 all employ CXSTACK 1401. In addition, execution of all of the program components by DEV CTL 1701 is controlled by variables in CVs 1411.

13. Operation of the Presentation Apparatus: Figs. 12-17

As previously indicated, the Presentation Apparatus has three basic types of operations: authoring the presentation, running the presentation, and loading or storing the presentation. These types of operations are described in the following. Included in the discussion of authoring are discussions of the display of ISTRUC 505, specifying active icons and ranges of icons, the BUILD operation as a typical structure editing operation, and content editing for the GRAPHIC icon.

<u>13a. Displaying ISTRUC 505</u>

In displaying ISTRUC 505, the problem is to determine what portion of ISTRUC 505 is to be displayed on SCR 121 at a given time. The problem is solved in the preferred embodiment by defining an "icon space" in which the position of every INODE 1205 is given by means of an X and a Y component for the INODE 1205. As indicated in the discussion of node 1203, DISPD field 1333 of node 1203 representing an INODE 1205 contains YCOORD field 1337 and XCOORD field 1339 specifying the INODE 1205's X and Y coordinates in "icon space".

In displaying ISTRUC 505, SDISP 1705 determines from the current value of ICON SIZE 209, stored in ISIZE 1429 of control variables 1411, and the size of SCR 121, stored in SCR WIDTH 1417 and SCR HEIGHT 1419, how much of the "icon space" can presently be displayed on SCR 121. Next, it determines SCR 121's present location relative to "icon space", which it indicates by means of SCRX 1421 and SCRY 1423 specifying the lower right-hand corner of SCR 121 in "icon space" and X and Y coordinates computed therefrom to indicate the upper left-hand corner.

The first INODE 1205 to be processed for display is located by the pointer stored in REF_PTR 1435. REF_PTR 1435 is reset to HEAD_PTR 1431 each time SCR 121 is scrolled up or to the left in icon space. Beginning with the INODE 1205 specified by REF_PTR 1435, SDISP 1705 continues processing INODEs 1205 until it reaches an INODE 1205 whose YCOORD field 1337 and XCOORD field 1339 indicate that it is below or to the right of the portion of "icon space" displayed on SCR 121.

The first step in processing each INODE 1205 is to determine whether the current INODE 1205 is below or to the right of the displayed portion of "icon space"; if it is, SDISP 1705 is finished. Otherwise, SDISP 1705 checks whether the current INODE 1205 represents a node 203 within the range specified by S_RANGE PTR 1437 and E_RANGE_PTR 1439, which are set by the range editing operation. If it is, SDISP 1705 sets the IN RANGE flag in SFLAGS 1301 of INODE 1205. Next, SDISP 1705 determines from the INODE 1205's coordinates in DISPD 1333 and the present location of SCR 121 relative to "icon space" whether INODE 1205 is within the displayed portion of "icon space"; if it is, SDISP 1705 displays icon 203 corresponding to INODE 1205 using the information in SFLAGS 1301 to determine the icon 203's color, the information in ITYPE 1303 to provide IPIC 205, and the information in INAME 1305 to provide ILABEL 207. If the displayed INODE 1205 has a non-null DOWN PTR 1323, SDISP 1705 displays a vertical link 503; if it has a non-null RT PTR 1327 or a non-null CXPTR 1329, SDISP 1705 displays a horizontal link 503.

If REFPTR 205 is not in within the displayed portion and has only DOWNPTR 1323, the current INODE 1205 becomes the new REFPTR 205. If the current INODE 1205 is the last INODE 1205 in the range, there is no further need to change the color of nodes 203 representing INODEs 1205 and the display operation ceases.

Otherwise, SDISP 1705 checks the current INODE 1205 for a non-null RT PTR 1327, a non-null DOWN PTR 1323, or a non-null CXPTR 1329; if the INODE 1205 has none of these or is a LOOP END INODE 1205, SDISP 1705 is finished. Otherwise, if the current INODE 1205 specifies a branch, i.e., if it has both a non-null DOWN PTR 1323 and either a non-null RT PTR 1327 or a non-null CXPTR 1329, then SDISP 1705 determines which of the branches is to be represented by an ARROW icon 1103. If there is an INODE 1205 at the location referred to by DOWN PTR 1323, the INODE 1205 specified by the DOWN PTR 1323 is represented by a DOWN ARROW icon 1103, the INODE 1205 pointed to by RT PTR 2317 or CXPTR 1329 becomes the new current INODE 1205, and the loop is repeated. If there is no INODE 1205 at the location referred to by DOWN PTR 1323, the INODE 1205 specified by RT PTR 1327 is represented by a RIGHT ARROW icon 1103 and DOWN PTR 1327 is followed to the next INODE 1205. If the current INODE 1205 only has one of DOWN PTR 1323, RT PTR 1327, or CXPTR 1329, the INODE 1205 specified by that pointer becomes the new current node and the loop is repeated.

When the author uses scroll bar 213 to relocate SCR 121 relative to "icon space", SDISP 1705 responds by recomputing the location of SCR 121 in "icon space", setting REF_PTR 1435 to HEAD_PTR 1431, and continuing as just described. Similarly, when the author clicks on HOME 217, SDISP 1705 responds by locating SCR 121 in the upper left hand corner of "icon space" and using HEAD_PTR 1431 as REF_PTR 1435.

Display of ILIB 303 by SDISP 1705 is similar. SDISP 1705 maps the library window onto the "library space" occupied by library structure 1601. The topmost socket CXNODE 1605 currently within the library window is indicated by LIBREF PTR 1449. SDISP 1705 walks down LIBLIST 1603, displaying the icon 203 corresponding to SINODE or CXNODE 1206 immediately attached to socket CXNODE 1605. In other embodiments, SDISP 1705 may perform selective display of CXPSTRUC 1607 for a CXNODE 1207 in library structure 1601.

As described in the discussion of ARROW icons 1103, when the author clicks on an ARROW icon 1103, a portion of the branch specified by ARROW icon 1103 is displayed and the other branch is represented by an ARROW icon 1103. SDISP 1705 performs this operation by resetting ARROWFL 1335 in the first SINODE 1205 of for the branch being displayed, setting ARROWFL 1335 in the first SINODE 1205 of the other branch, moving the location of SCR 121 in "icon space" so that the branch can be displayed, and continuing with the display operation as described above.

13b. Selecting an Operation

In presentation apparatus which permits editing, SDISP 1705 displays ISTRUC 505 as described above upon termination of each execution of SED 1702, CED 1703, SAVE/LOAD 1711, and RUN 1709. As previously described, the operations are selected by clicking on menu bar 211 and then on the resulting pull down and pull right windows. When a specific operation is chosen, a flag in STATE FLAGS 1413 is set and the value of that flag determines whether SED 1702, CED 1703, RUN 1709, or SAVE/LOAD 1711 will next be executed.

13c. Selecting icons 203 and specifying ranges

As previously explained, when an author is editing a presentation, he selects an icon 203 by clicking on it with mouse 109. In a preferred embodiment, when the author clicks on the icon 203, DEV CTL 1701 receives an input from DIN 119 which it converts to a pair of values specifying coordinates for the current location of C 111 in SCR 121. SDISP 1705 receives those values and converts the screen coordinates to icon space coordinates. If there is an INODE 1205 at that location in icon space, SDISP 1705 sets SFLAGS 1301 to indicate that the icon is active. The next time SDISP 1705 displays ISTRUC 505, the corresponding icon 203 will be colored purple. Activation of a TICON in LSTRUC 1601 is similar. When a range is specified, the procedure is the same except that a pointer to INODE 1205 corresponding to icon 203 at the beginning of the range is stored in S_RANGE_PTR 1437 and a pointer to INODE 1205 corresponding to icon 203 at the end of the range is stored in E_RANGE_PTR 1437. As described in the discussion of display operations, SDISP 1705 sets the IN_RANGE flag in SFLAGS 1301 for all INODES 1205 in the range to indicate that the corresponding icon 203 is active.

13d. Structure Editing

The BUILD operation may serve as an example of the structure editing operations. As previously stated, an author performs the BUILD operation on a previously-selected TICON 305 in ILIB 303 which represents the SICON 902 or CXICON 803 is to be inserted into ISTRUC 505. The author specifies the BUILD operation and the build direction on the pull-down and pull-right windows. In response to those inputs, SED 1702 first walks LSTRUC 1601 until it locates socket CXNODE 1605 in LSTRUC 1601 whose SFLAGS 1301 indicate that it has been activated and copies a pointer to the INODE 1208 dependent from that socket CXNODE 1605 into LIB B PTR 1446 of CVs 1411. Then SED 1702 recursively walks PSTRUC 1201 in the manner described for SDISP 1705 until it locates INODE 1205 whose SFLAGS 1301 indicate that it has been activated SED 1702 assigns the location of that INODE 1205 to S_RANGE PTR 1437. Thereupon, SED 1702 determines from the author's inputs whether the new nodes 1203 are to be attached to DOWN PTR 1323 or RT PTR 1327 of the activated INODE 1205. SED 1702 then makes a copy of the nodes 1203 pointed to by LIB B PTR 1446 and links them into PSTRUC 1201 at DOWN PTR 1323 or RT PTR 1327. Finally, SED 1702 recursively walks the modified PSTRUC 1201 from the active node to its end and sets the values of YCOORD 1337 and XCOORD 1339 as required to reflect the changes in "icon space" resulting from the addition of the new nodes to PSTRUC 1201.

The COPY, CUT, and PASTE operations work in the manner described above for BUILD, except that the nodes 1203 involved in the operation are copied into PBUFF 1723. In the case of CUT, the nodes 1203 selected for the CUT operation are removed from PSTRUC 1201 and PSTRUC 1201 is relinked after they are copied; in the case of COPY, the nodes 1203 are simply copied into PBUFF 1703. In the PASTE operation, a copy of the current contents of PBUFF 1703 are added to PSTRUC 1201 at the location specified for the operation. Similarly, in the COMPOSE operation, a new socket CXNODE 1605 is added at the proper point in LISRUC 1601 and the portion of PSTRUC 1201 which defines the new complex icon is copied into LSTRUC 1601 and linked to the new socket CXNODE 1605 and in the DECOMPOSE operation, a socket CXNODE 1605 and its dependent nodes 203 are removed from LSTRUC 1601.

13e. Content Editing

As previously explained, content editing may be done on a single icon 203 or a range of icons 203. CED 1703 performs content editing by recursively walking PSTRUC 1201 over the range indicated in the manner described for SDISP 1705 and adding or modifying CNODEs 1207. For each SINODE 1208, CED 1703 determines from ITYPE 1303 what kind of icon 203 SINODE 1208 represents and invokes a routing which provides the screens necessary to receive input for the type, creates CNODEs 1207 to receive the input, and links them to the SINODE 1208. For example, if the SINODE 1208 whose content is being edited is the SINODE 1208 corresponding to the GRAPHIC icon 203, the routine causes DEV CTL 1701 to output screens to which the user responds by providing a value which resolves to the name of a graphics file and a value which resolves to the file's types, creates the two CNODES 1207 used with the GRAPHIC INODE 1205 to hold the information, and links them via CLINKs 1211 to the graphic SINODE 1208. When content has been added, CED 1703 sets the CONTENT flag in SFLAGs 1301, to which SDISP 1705 responds by coloring icon 203 corresponding to SINODE 1208 yellow the next time it displays ISTRUC 505.

13f. Running a Presentation

When the author clicks on the "control" field in menu bar 211 and then on the "run" field in the pull-down menu, RUN 1709 begins executing PSTRUC 1201 from the first node 203 therein, indicated by HEAD_PTR 1431. RUN is implemented as a loop which continues executing as long as CURR_PTR 1435, which contains the pointer to the INODE 1205 presently being executed, is non-null. When CURR PTR 1435 is null, the loop terminates the RUN returns. On each iteration of the loop, RUN uses CURR_PTR to locate PROC PTR 1311 in the current INODE 1205. RUN then executes the procedure specified by PROC PTR 1311. The procedure always returns a pointer to the next INODE 1205 to be, and the returned pointer is assigned to CURR_PTR. Where no branch is involved, the returned pointer is always DOWN PTR 1323 from the INODE 1205 being executed. If the INODE 1205 has content, the procedure performs the operation specified by the INODE 1205 using the content of the CNODEs 1207 dependent from it before returning DOWN PTR 1323.

With branches, RUN proceeds as follows: in the case of a CXNODE 1207, the procedure for that node type saves DOWN PTR 1323 in CXSTACK 1401 and recursively invokes RUN using CXPTR 1329. On return from the recursion, the procedure returns DOWN PTR 1323 as the pointer to the next INODE 1205. In the case of a LOOP CXNODE 1207, the LOOP END SINODE 1208 has a pointer to the LOOP START SINODE 1208 in RT PTR 1327 and the procedure for the LOOP END SINODE 1208 simply returns that pointer. If a LOOP EXIT SINODE 1208 is executed during an iteration of the loop, the procedure for LOOP EXIT sets a flag in LOOP FLAGS 1415 and RUN responds to the set flag by setting CURR PTR 1433 to null, terminating RUN's loop and its recursive invocation. In the case of an IF SINODE 1208 with content, the pointer returned by the procedure depends on the value of the content.

13g. Saving and Loading

When saving a PSTRUC 1201 or a LSTRUC 1601, SAVE/LOAD makes a sequential representation of PSTRUC 1201 or LSTRUC 1601 in which each node 1203 has an index number, CLINKs 1211 and SLINKs 1209 are replaced by the index numbers of the nodes 1203 the pointers point to, and PPTRs 1210 are replaced by symbolic references to the procedures. When the sequential representation is complete, it is written to a disk file. When saving VARLIST 1501, SAVE/LOAD nakes a record from each VLE 1503 which contains VNAME 1505, VTYPE 1507, VVAL 1509, and sequence number of the next record to be written in NPTR 1511. The record is written to a disk file. Further, If there is a pointer to the VLE 1503 in HT 1521, SAVE/LOAD replaces the pointer with the sequence number of the VLE's record. HT 1521, too, is then saved in a disk file.

When loading a saved PSTRUC 1201 or LSTRUC 1601, SAVE/LOAD performs the reverse conversion. The nodes 1203 in the file are read out in sequence and as each node is read out, it is placed in its proper loacation in PSTRUC 1201 or LSTRUC 1601, the index numbers are replaced with pointers, and the symbolic procedure references are replaced with procedure pointers. Similarly, when VL 1501 is loaded, the sequence in the records and HT 1521 are replaced with pointers.

The conversions on saving and loading free PSTRUC 1201, LSTRUC 1601, and VARLIST 1501 from dependence on any particular memory configuration, and thus permit transfer of presentations between CSs 105, but at the same time provide for rapid execution of operations involving PSTRUC 1201, LSTRUC 1601, and VARLIST 1501. In a preferred embodiment, the conversions involving PSTRUC 1201 and LSTRUC 1601 are performed using recursive routines.

14. Discussion of Types of SINODEs 1208 and SICONs 902 in a Preferred Embodiment

The SINODEs 1208 and their corresponding SICONS 902 in a preferred embodiment fall into two classes: those that have general programming functions and would be useful in any application employing the authoring interface described herein and those that have functions specific to presentations. The members of both classes of SICONs 902 and their SINODEs 1208 employed in a preferred embodiment will be priefly described in the following. It is to be understood in the discussion that the content of a SICON 902 may in general be a literal value or the contents of a variable. In other embodiments, other expressions may be permitted. As indicated in the discussion of the GRAPHIC SICON 902, the values may be defined interactively.

14a. Specialized Presentation SICONs 902

Since the specialized presentation SICONs 902 can be understood by one skilled in the presentation art in light of the description of the GRAPHIC SICON 902 and its SINODE 1208 above, they will be summarily presented as follows:

The AUDIO SICON 902 controls one of the two audio channels provided by VDB 101 to MMGW 103. The icon's contents are the channel number and whether the channel is on or off.

the BOX SICON 902 describes a box to be drawn on MMGE 103; the contents are the box's coordinates, the background color, and the color of the box outline.

the CIRCLE SICON 902 describes a circle to be drawn on MMGE 103; the contents are the circle's center coordinates, its radius, the background color, and the color of the circle outline.

the CLEAR SICON 902 clears SCR 121; in a preferred embodiment, its content is the color of the background of the authoring screen.

the FIND SICON 902 specifies that VDB 101 locate a specific frame in the video disk. The content is the frame number, which may be defined interactively via the video editor.

the GRAPHIC SICON specifies a graphic to be displayed on SCR 121; it has already been described in detail.

The INPUT SICON 902 specifies that the presentation is to receive input from mouse 109 or KB 107. The content is values which resolve to the variable name to receive the input, the type of input, whether the input should be displayed, and how long the presentation should wait for input before proceeding.

The LINE SICON 902 specifies that a line is to be displayed. The content is values which resolve to start and end coordinates for the line and the line's color.

The MODE SICON 902 specifies one of three display modes for MMGE 103: graphics only, video only, or graphics superimposed on video. The content is a value which resolves to a specification of one of the three modes.

The PAUSE SICON 902 specifies that the presentation whould pause in its present state for a specified time. The content is a value which resolves to the time in seconds.

The PLAY SICON 902 specifies that a section of the videodisc beginning at a location previously specified in the FIND SICON 902 be played. The content is a value which resolves to the frame at which the section to be played ends. Again, the value may be determined interactively.

The SHOWVAR SICON 902 specifies that the value of a variable be displayed on MMGW 103. The contents are values which specify the variable, the foreground and background colors, the size of the display, and the location of the display on the screen.

The TEXT SICON 902 specifies text from the text file to be displayed on MMGW 103. The contents are values which specify the text file, the foreground and background colors, the size of the text in the display, and the location on the screen.

The WRITE SICON 902 specifies a text string to be displayed on MMGW 103. The contents are values which specify the string, the foreground and background colors, the text size, and the location of the text on the screen.

Where the content of one of the above SICONs 902 requires specification of size, color, or location, the content editor provides the author with interactive editors which permit him to specify size, color, and location by specifying them directly on a display screen.

14b. SICONs 902 with General Programming Functions

In the following, SICONs 902 with general programming functions and their corresponding SINODEs 1208 are discussed in more detail. In a preferred embodiment, the SICONs 902 are the following:

ASSIGN, which assigns a value to a variable;

DETOUR, which responds to a specified interrupt condition by invoking another program, executing it, and returning.

EXIT, which exits a loop, a complex icon, or a course.

IF, which selects a branch on the basis of a condition.

LOAD, which loads a program.

LOADVAR, which loads a variable list.

SAVE, which saves a program.

SAVEVAR, saves a variable list.

Additionally, there are two built-in complex icons for programming structures: LOOP and BRANCHES. The icons are discussed in the above order.

The ASSIGN SICON 902 permits the author to define a variable and an expression by means of which the variable's value is computed. The expression may be a simple expression specifying a constant or another variable, or it may be a complex expression involving constants, variables, and one or more operators. In a preferred embodiment, the operators are the following:

Operator	Operation
+	addition
-	subtraction
*	multiplication
/	division

When specifying the content of ASSIGN SICON 902, the author specifies the variable's name, which must begin with the character "@", and the expression, which contains variable names or constants and is written in the usual arithmetic fashion. For example

@var_1 + 2

would specify that the constant 2 be added to the present value of var_1 and the result assigned to the variable specified in the ASSIGN SICON 902. In a preferred embodiment, when a variable is defined for an ASSIGN icon, as many characters of the variable's name as will fit on the ASSIGN SICON 902 appear on the SICON 902.

Fig. 18 shows a detail of ASSIGN SINODE 1208 corresponding to ASSIGN SICON 902. CLINKs 1211 of ASSIGN SINODE 1801 point to CNODE 1207 VAR 1803, which contains the name of the variable to which the expression is to be assigned and to a parse tree 1809 of CNODEs 1207 which represent the expression. The parse tree for the example expression above is shown in Fig. 18: OP 1805 specifies the addition operation. OP 1805 is a binary operation whose operands in the present example are simple expressions, and consequently, there are two further CNODEs 1207 dependent from OP 1805, one representing @var_1 and the other representing the constant 2. EPROC PTR 1313 in OP 1805 points to the function which performs the addition and returns the result. When RUN executes an ASSIGN SINODE 1801, the procedure specified in the SINODE 1801 recursively executes the tree of CNODEs 1207 dependent from SINODE 1801. In the course of the recursions, @var_1's entry in VARLIST 1501 is located and @var_1's value retrieved from VVAL 1509, the function specified by EPROC PTR 1313 or OP 1805 adds @VAR_1's value and 2, and the result is written into VLE 1503 for the variable to which the expression is being assigned.

The DETOUR SICON 902 defines an interrupt condition, a program to be executed when the interrupt occurs, and whether the interrupt condition is on or off. The conditions which may cause the interrupt include in a preferred embodiment signals from the buttons on mouse 109 and input from KB 107. When a condition specified in a DETOUR SICON 902 occurs at any point in the program between the point where one DETOUR SICON 902 turns the condition on and another turns it off, the program is immediately executed. When execution of the interrupt program is over, execution of the interrupted program continues from the point where the interrupt occurred. The content for DETOUR SICON 902 include the interrupt condition, the file name of the program to be executed, and whether the condition is on or off.

The EXIT SICON 902 specifies an exit from the complex icon, loop, or program to which the EXIT SICON 902 immediately belongs. The exit occurs when the SINODE 1207 corresponding to the EXIT SICON 902 is executed and terminates execution of the structure from which the exit occurs. For example, if a loop contains a CXBR 1223, a complex brancy EXIT SINODE 1207 in the CXBR 1221 terminates execution of CXBR 1223, but not the execution of the loop and of every structure contained therein. If the execution of a loop, complex icon, or program is nested in the execution of another loop, complex icon or program, only the execution of the complex branch, loop, complex icon, or program to which the EXIT SINODE 1207 immediately belongs is terminated. The content of EXIT SINODE 1207 is the type of exit.

The IF SICON 902 permits the author to define a condition and a branch to be taken if the condition is true. In defining the content of an IF SICON 902, the author specifies two values to be compared (for example, two variables or a variable and a constant), the data type of the values, and the manner in which they must compare if the IF SICON 902 is to test true. In a preferred embodiment, the tests are EQUAL, NOT EQUAL, LESS THAN and GREATER THAN. A detail of the IF SINODE 1208 appears in Fig. 12. As may be seen there, one tree of CNODEs 1207 specifies the values to be compared and the kind of comparison; the other CNODE 1207 specifies the data type for the test to be performed. The routing that performs the actual evaluation is pointed to by EPROC PTR 1313.

The LOAD SICON 902 specifies that another program be loaded and executed and that after execution is finished, execution of the program containing the LOAD SICON 902 continues with the INODE 1205 following the LOAD SICON 902. The content of a LOAD SICON 902 is the file name of the program to be loaded. The LOADVAR SICON 902 specifies that the variables saved in a file be added to VARLIST 1501. In a preferred embodiment, if a saved variable has the same name as one already on VARLIST 1501, the VLE 1503 from the file replaces the VLE 1503 for that variable in VARLIST 1501; in other embodiments, the content of the LOADVAR SICON 902 may permit the user to specify whether identical variables are to be replaced. Again the content of the LOADVAR SICON 902 is the name of the file.

The SAVE AND SAVEVAR SICONs 902 specify that the program currently being executed and the current variable list respectively be saved on disk. The content of the icons is the name of the file to which they are to be saved.

The LOOP built-in CXNODE 1207 corresponding to the LOOP CXICON *03 is illustrated in Fig. 12. The components of the LOOP CXNODE 1207 are the L-START SINODE 1208, whose procedure simply provides its DOWN PTR 1323 to RUN, and the L-END SINODE 1208, whose procedure provides RT PTR 1327 to the RUN. In the L-END SINODE 1208, that pointer points to the L-START SINODE 1208 for the loop. As may be seen from the above, loops in the preferred embodiment execute until a LOOP EXIT SINODE 1208 is executed. Of course, the ASSIGN SINODE 1208 and the IF SINODE 1208 may be employed in the LOOP CXNODE to produce the usual variations on the loop construct.

Though built-in in a preferred embodiment, the CHOICES CXICON 803 is an example of how the programming primitives just described can be used to construct CXICONs 803 and their corresponding CXNODEs 1207 which provide high-level programming functionality. As shown in Fig. 12, the CHOICES CXNODE 1207 provides a sequence of IF SINODEs 1208 in which each IF SINODE 1208 is the right branch of the preceding IF SINODE 1208. The down branch of course contains the nodes 1203 1203 which will be executed if the IF condition is true. The CHOICES CXNODE 1207 thus provides the functionality of the IF -- ELSEIF statements of some high-level programming languages and may also be used to construct a case statement.

15. Conclusion

The foregoing Description of a Preferred Embodiment has disclosed how one may construct a visual programming system in which the program is represented by a program structure of linked nodes representing steps in the program and their order of execution and the structure is interpreted both to produce a representation of the program on a display device wherein icons represent steps in the program and interconnections between the icons specify the order in which the steps are executed and to execute the program. Further included in the disclosure have been characteristics of the visual programming system such as the manner in which branches in the program structure are executed, displayed, defined, and added to a program structure, separate structure and content editing, the use of the appearance of an icon to indicate the status of the icon in the program writing process, the manner in which an icon library is used and added to, and the manner in which certain of the nodes in the program structure are executed. Finally, the disclosure has shown how the visual programming system may be employed in an apparatus for constructing and executing presentations.

While the visual programming system is described as it has been implemented in the presentation apparatus, the visual programming system is by no means limited to this application but may be used to create programs in any area. Moreover, the visual programming system is not limited to the hardware and computer system employed to implement the preferred embodiment, but may be used on any computer system which includes display apparatus having graphics capabilities. While color display apparatus is particularly advantageous, the invention may also be emodied in systems employing monochrome display apparatus. In such systems, means other than color may be used to distinguish icon types and conditions. Other peripherals used in the computer system will depend on the area of application of the visual programming system. In particular, the visual programming system may be implemented using a mouse with a different user interface or with other types of pointing devices. Further, the visual programming system may be implemented in a computer system with windowing capabilities. In that case, the icon library, the icon structure and the program execution could all be displayed in separate windows, and the author could thus simultaneously view the icon structure and the results of executing the program which it represents. Further, the general principles of the visual programming system disclosed herein may be applied in other systems using icons having forms, color, and representations of interconnections different from those described herein.

The embodiments disclosed herein are thus purely illustrative and exemplary and the scope of the invention is not limited by the embodiments, but is instead determined solely by the appended claims and includes all embodiments which come within the meaning and range of equivalency of the claims.

What is claimed is:

CLAIMS

1. Apparatus for iconographically representing and executing a program in a computer system including memory and a display terminal comprising:
   a program structure in the memory in which step nodes represent steps in the program and links between the step nodes specify the sequence of execution of the steps;
   display means responsive to the program structure for causing the display terminal to display a representation of the program structure including interconnected icons, the icons representing program steps and the interconnections the order of execution of the steps; and
   program execution means responsive to the program structure for causing the computer system to execute the steps represented by the step nodes in the order specified by the links.

2. The apparatus of claim 1 wherein:
   the computer system further includes input means associated with the display terminal for receiving command input including a run command and
   the program execution means is additionally responsive to the run command and responds to the program structure in response to the run command.

3. The apparatus of claim 2 and wherein:

the command input further includes a set of editing commands and the apparatus further comprises editing means responsive to the editing commands for modifying the program structure as specified by the editing commands.

4. The apparatus of claim 3 wherein:

the program structure further contains content nodes linked to certain of the step nodes which contain values used to execute the steps represented by the step nodes to which they are linked; and

the program execution means executes the step nodes regardless of whether there are content nodes linked thereto and employs the values in the content nodes in the execution of the step nodes to which the content nodes are linked.

5. The apparatus of claim 4 wherein:

the editing commands include

structure editing commands to which the editing means responds by modifying the links between the step nodes and

content editing commands to which the editing means reponds by modifying the content nodes.

6. The apparatus of claim 4 and wherein:

an icon corresponding to a step node which has a link to a content node is marked to indicate that values have been associated with the corresponding step node.

7. The apparatus of claim 3 and wherein:
the command input further includes an icon library display command;
the apparatus further comprises a library structure containing nodes representing the types of icons available for use in programs;
the display means responds to the icon library display command by displaying a representation of the library structure;
the editing commands include a build command specifying an icon of the representation of the library structure and a location in the representation of the program structure; and
the editing means responds to the build command by placing a node specifying the specified type of icon in the program structure at the location corresponding to the specified location.

8. The apparatus of claim 7 and wherein:
the step nodes include a branching step node which introduces a branch and which has a first link to the first step node of the branch and a second link to the first step node following the branch;
the types of icons represented in the library structure include a branching icon representing the branching step node the branch attached thereto;
the editing means responds to a build command specifying a branching icon be placing the branching step node and the represented

branch at the specified location in the program structure; and

the program execution means responds to the branching step node by executing the step nodes in the branch until execution of the branch is complete and then continuing execution with the first step node following the branch.

9. The apparatus of claim 8 and wherein:

the editing commands include a compose command specifying a sequence of icons in the representation of the program; and

the editing means responds to the compose command by creating a branching step node in the library structure which represents the specified sequence of step nodes.

10. The apparatus of claim 1 and wherein:

the step nodes include a branching step node which introduces a branch and which has a first link to the first step node of the brancyh and a second link to the first step node following the branch and

the program execution means responds to the branching step node by executing the step nodes in the branch until execution of the branch is complete and then continuing execution with the first step node following the branch.

11. The apparatus of claim 1 and wherein:

the step nodes include a load step node which specifies a second program structure; and

the program execution means responds to the load step node by executing the second program structure and then continuing execution with the first step node following the load step node.

12. Presentation apparatus for use in a computer system including memory and presentation output means comprising:

a presentation structure in the memory in which nodes represent steps in the presentation and links between the nodes specify the sequence of execution of the steps; and

presentation execution means responsive to the presentation struction for causing the presentation output means to execute the presentation specified by the presentation structure.

13. The presentation apparatus of claim 12 and further comprising:

interactive input means associated with the presentation output means;

a branching node in the presentation structure which specifies a branch in the sequence of execution in response to input from the interactive input means;

and wherein the presentation execution means responds to the branching node and the input by performing the specified branch as determined by the branching node and the input.

14. The presentation apparatus of claim 12 and further comprising:
presentation structure display means responsive to the presentation structure for causing the presentation output means to display a representation of the presentation structure including interconnected icons representing the nodes and the links between the nodes.

15. The presentation apparatus of claim 14 and further comprising:
interactive input means associated with the presentation output means for receiving commands including editing commands; and
the apparatus further comprises editing means responsive to the editing commands for modifying the presentation structure as specified by the editing commands.

16. In a computer system having input means and a graphics display device,
apparatus for writing a program comprising:
an iconic representation of the program on the display device as a structure of icons and interconnections wherein the icons represent steps in the program including program branches, the structure represents the order in which the steps are executed, and a given icon has either a single first interconnection to an immediately preceding icon representing any step or a single second interconnection which

is visually distinct from the first interconnection to an immediately preceding icon representing a branching step; and

iconic representation manipulation means responsive to the input means for manipulating the iconic representation and producing an executable representation of the program corresponding to the iconic representation.

17. In the program writing apparatus of claim 16 and wherein:

the icons representing branching steps include a first icon specifying that the branch beginning at the first icon be completely executed and thereupon that the execution continue with the icon whose first interconnection is to the first icon.

18. In the program writing apparatus of claim 17 and wherein:

the branch beginning at the first icon is unconditional.

19. In the program writing apparatus of claim 17 and wherein:

the icons representing branching steps include a second icon specifying a conditional branch.

20. In the program writing apparatus of claim 16 and wherein:

the first interconnection is vertical and the second is horizontal.

21. In a computer system having input means and a graphics display device,

    apparatus for writing a program comprising:

    an iconic representation of the program on the display device as a structure of icons where the icons represents steps in the program, the structure represents the order in which the steps are executed, and the structure includes a branch specifying that the steps represented by the icons in the branch be executed until the end of the branch is reached and that the execution continue thereafter with the step represented by the icon following the point at which the branch begins; and

    iconic representation manipulation means responsive to the input means for manipulating the iconic representation and producing an executable representation of the program corresponding to the iconic representation.

22. In the program writing apparatus of claim 21 and wherein:

    the branch is introduced by an icon having an appearance different from icons which do not introduce the branch.

23. In the program writing apparatus of claim 22 and wherein:

    the graphics display devide is a color device; and the branch introducing icon has a color different from that of icons which do not introduce the branch.

24. In the program writing apparatus of claim 21 and wherein:
the icons include a branch exit icon specifying an exit step which terminates execution of the branch prior to the end of the branch.

25. In the program writing apparatus of claim 21 and wherein:
the icons include a loop start icon and a loop end icon specifying steps which define a loop in the branch.

26. In the program writing apparatus of claim 25 and wherein:
the icons further include a loop exit icon specifying an exit step which terminates execution of the loop and the branch in which the loop is defined.

27. In the program writing apparatus of claim 21 and wherein:
the apparatus further includes a library containing the branch; and
the iconic representation manipulation means includes means for copying the branch from the library to a specified point in the structure.

28. In the program writing apparatus of claim 27 and wherein:

the branch is visually represented in the library by a branch icon; and

the means for copying the branch selects the branch icon and the specified point; and

the iconic representation manipulation means add the branch icon and the icons representing the steps of the branch to the structure at the specified point.

29. In the program writing apparatus of claim 28 and wherein:

the input means includes a pointer device; and

the branch icon and the point are selected by means of the pointer device.

30. In the program writing apparatus of claim 27 and wherein:

the iconic representation manipulation means includes means for defining a sequence of the icons in the structure as the icons belonging to a branch to be added to the library and adding the branch to the library.

31. In the program writing apparatus of claim 30 and wherein:

the iconic representation manipulation means further includes means for defining the branch icon representing the branch to be added.

32. In a computer system having input means and a graphics display device,

apparatus for writing a program comprising:

an iconic representation of the program on the display device as a branched structure of icons wherein the icons represent steps in the program, the structure represents the order in which the steps are executed, and the icons include an arrow icon indicating the direction of a branch and the kind of step represented by the first icon in the branch; and

iconic representation manipulation means responsive to the input means for manipulating the iconic representation and producing an executable representation of the program corresponding to the iconic representation.

33. In the program writing apparatus of claim 32 and wherein:

the arrow icon may be specified by means of the input means; and

the iconic representation manipulation means responds to the specification of the arrow icon by displaying a portion of the branch indicated by the arrow icon.

34. In the program writing apparatus of claim 33 and wherein:

the input means includes a pointer device; and

the arrow icon is specified by means of the pointer device.

35. In the program writing apparatus of claim 32 and wherein:
the graphics display device is a color display device; and
the arrow icon has a color different from that of other icons.

36. In a computer system having input means and a graphics display device,
apparatus for writing a program comprising:
an iconic representation of the program on the display device as a structure of icons representing program steps wherein an icon's appearance indicates the status of the represented program step while the program is being written; and
iconic representation manipulation means responsive to the input means for manipulation means responsive to the input means for manipulating the iconic representation and producing an executable representation of the program corresponding to the iconic representation.

37. In the program writing apparatus of claim 36 and wherein:
the graphics display device is a color device; and
the icon's color indicates the represented program step's status while the program is being written.

38. In the program writing apparatus of claim 36 and wherein:

the indicated status includes whether author-defined content has been associated with the represented program step.

39. In the program writing apparatus of claim 36 and wherein:

the indicated status includes whether the represented step has been selected for manipulation by the iconic representation manipulation means.

40. In a computer system having input means and a graphics display device,

apparatus for writing a program comprising:

an iconic representation of the program on the display device as a structure of icons which may have author-defined content; and

means producing an executable representation of the program corresponding to the iconic representation and including

structure editing means responsive to the input means for manipulating the structure of icons independently of the icons' content and

content editing means responsive to the input means for specifying the author-defined content for an icon in the structure.

41. In the program writing apparatus of claim 40 and wherein:

the icons have types indicating the classes of operations performed by the icons; and

the content editing means responds to the type of the icon for which author-defined content is being specified by permitting only specification of author-defined content corresponding to the type.

42. In the program writing apparatus of claim 40 and wherein:

an icon's appearance changes when the content editor specifies author-defined content for the icon.

43. In the program writing apparatus of claim 42 and wherein:

the graphics display device is a color device; and

the icon's color indicates whether the icon has author-defined content.

44. In the program writing apparatus of claim 40 and wherein:

the means for producing an executable representation of the program produces the executable representation even when no author-defined content has been specified.

45. Apparatus for executing a program in a computer system including memory means comprising:

a program structure representing the program in the memory means, the structure including nodes connected by links, the nodes including

steps nodes representing program steps and content nodes containing values and

the links including in each step node

a structure link specifying the step node representing the next program step and

a content link specifying a content node; and

means for executing the program represented by the structure by executing for each step node the program step represented by the step node and thereupon following a non-null structure link and executing the step node representing the next program step and if the step node has a non-null content link, employing the values in the content node in the execution of the step represented by the given step node.

46. In the apparatus for executing a program of claim 45 and wherein:

each step node includes a plurality of structure links, the structure links including

a next link indicating the next step node to be executed after execution of the current step node is complete and an unconditional branch link specifying the first step node in a branch of the structure; and

the program execution means follows a non-null unconditional branch link and executes the step nodes in the branch until it has completely executed the branch, whereupon it follows the non-null next link in the step node containing the non-null branch link.

47. In the apparatus for executing a program of claim 45 and wherein:
each step node includes two content links;
each content node includes two content links;
a step node with a non-null content link is a root node of a binary tree of content nodes; and
the program execution means walks the binary tree of content nodes to obtain the values employed in the execution of the step node.

48. In the program executing apparatus of claim 45 and wherein:
the computer system further includes graphic display means and input means;
the program executing apparatus further includes means for writing the program comprising
a representation of the program structure on the display means as a structure of icons and interconnections wherein each icon represents a step node and each interconnection a non-null structure link;
means for manipulating the program structure in response to the input means; and
means for interpreting the program structure to generate the representation of the program structure on the display device.

49. In a computer system having input means and a graphics display device,

apparatus for writing a presentation comprising;

an iconic representation of the presentation on the display device as a structure of icons and interconnections wherein the icons represent steps in the presentation and the structure represents the order in which the steps are executed; and

iconic representation manipulation means responsive to the input means for manipulating the iconic representation and producing an executable representation of the presentation corresponding to the iconic representation.

50. In the apparatus for writing a presentation of claim 49 and wherein:

the apparatus further includes a library containing icons; and

the iconic representation manipulation means includes means for copying an icon contained in the library to the structure.

51. In the presentation writing apparatus of claim 49 and wherein:

the icons include a macro icon representing a plurality of steps.

52. In the presentation writing apparatus of claim 51 and wherein:

the iconic representation manipulation means includes means permitting the author of the presentation to define the appearance of the macro icon.

53. In the presentation writing apparatus of claim 51 and wherein:

the apparatus further includes a library containing icons; and the iconic representation manipulation means includes means for defining a macro icon in the library and means for copying the defined macro icon to the structure.

54. In the presentation writing apparatus of claim 49 and wherein:

the icons include an icon representing a conditional branch.

55. In the presentation writing apparatus of claim 49 and wherein:

the icons include an icons representing a pause.

56. In the presentation writing apparatus of claim 49 and wherein:

the icons include an icon representing a graphics display.

57. In the presentation writing apparatus of claim 49 and wherein:

the icons include an icon representing the presentation of a choice to a user of the presentation and a response thereto.

58. In the presentation writing apparatus of claim 49 and wherein:

the iconic representation includes means for representing a loop in the presentation.

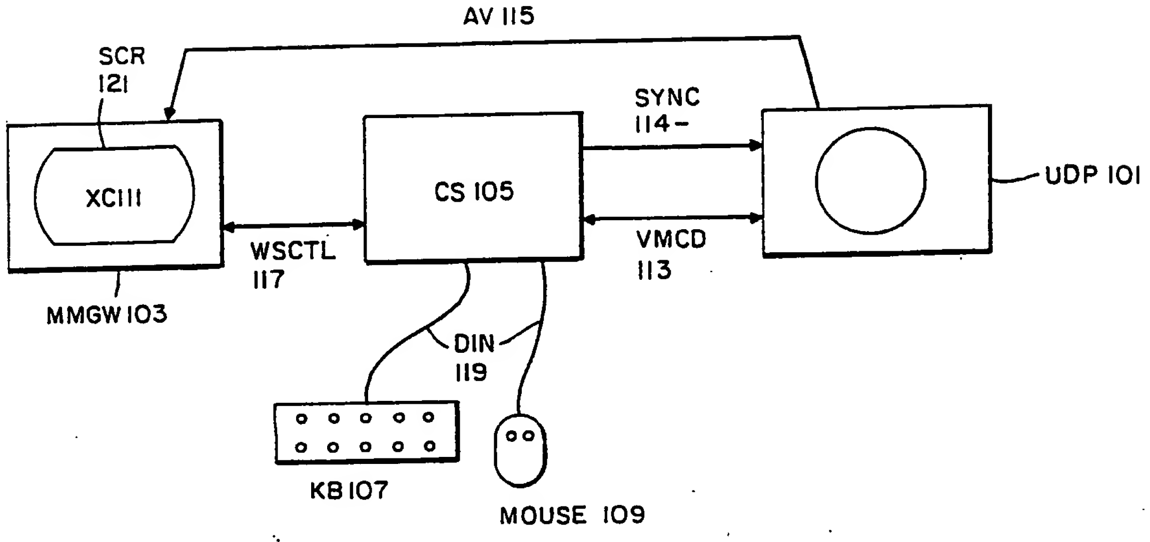


FIG. 1: PRESENTATION APPARATUS HARDWARE

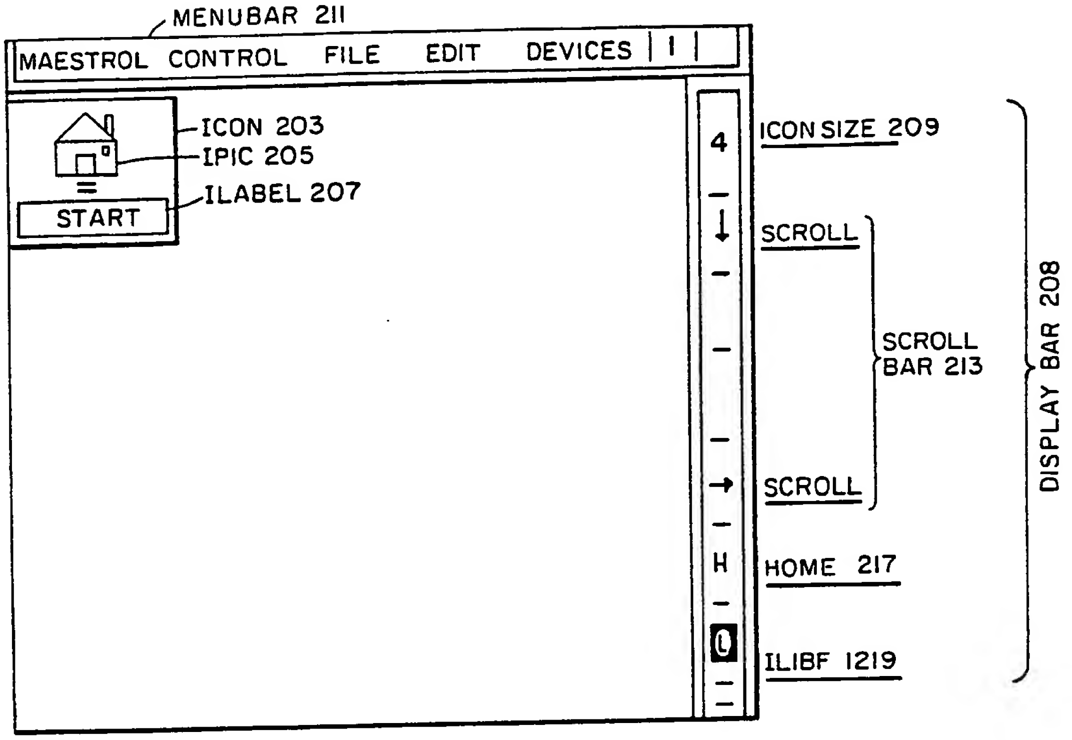


FIG. 2: ASCR(1) 201

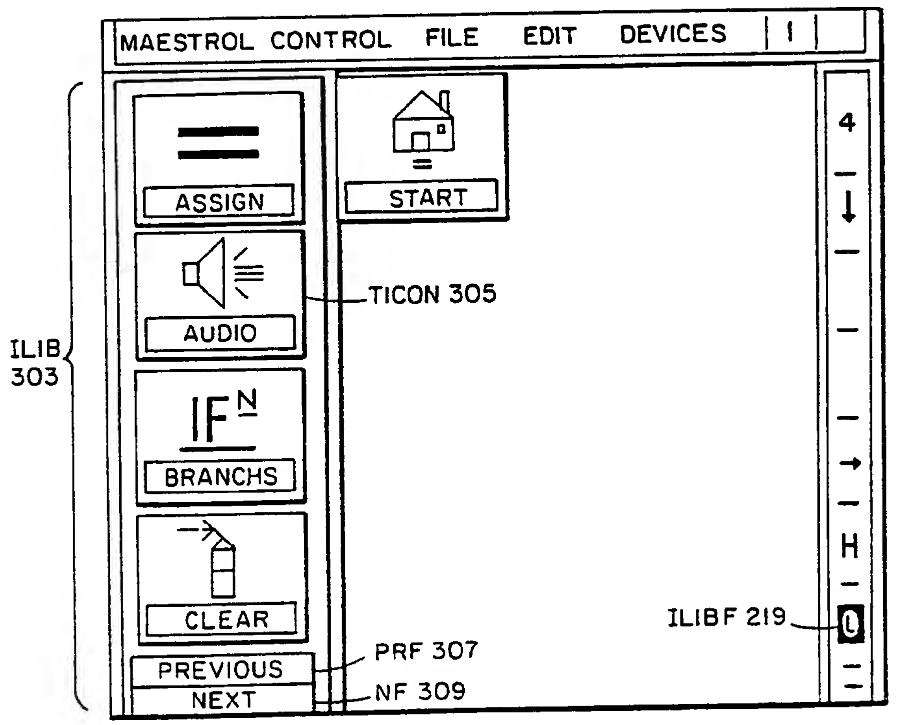


FIG. 3: ASCR(2) 301

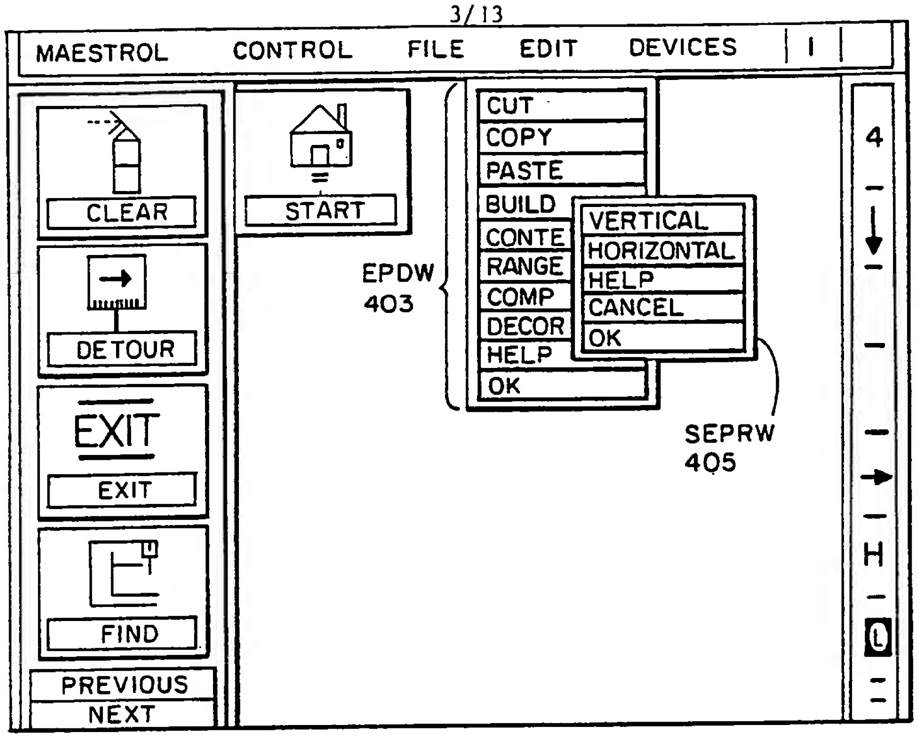


FIG. 4: ASCR(3) 401

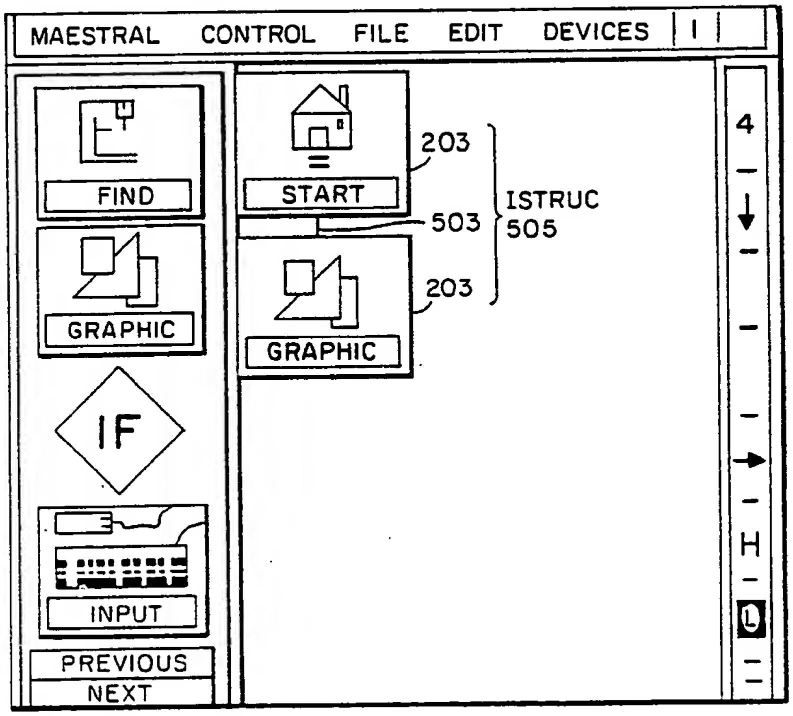


FIG. 5: ASCR(4) 501

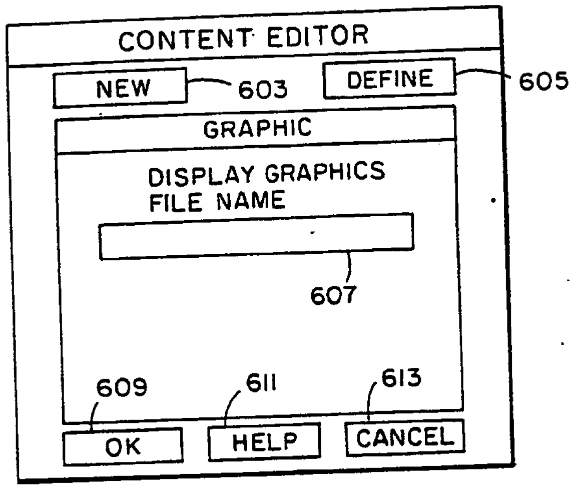


FIG. 6: CONTENT EDITOR WINDOW 601

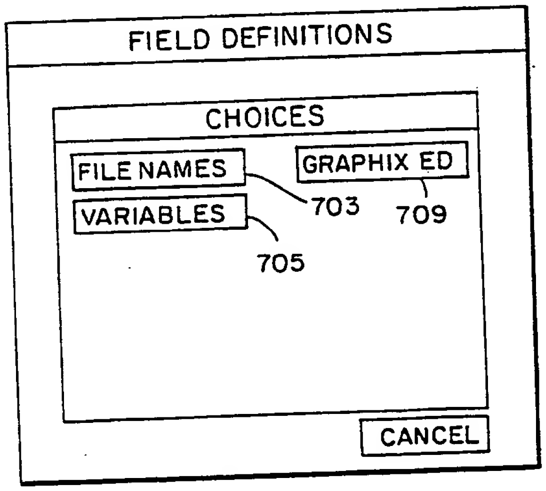


FIG. 7: DEFINE WINDOW 701

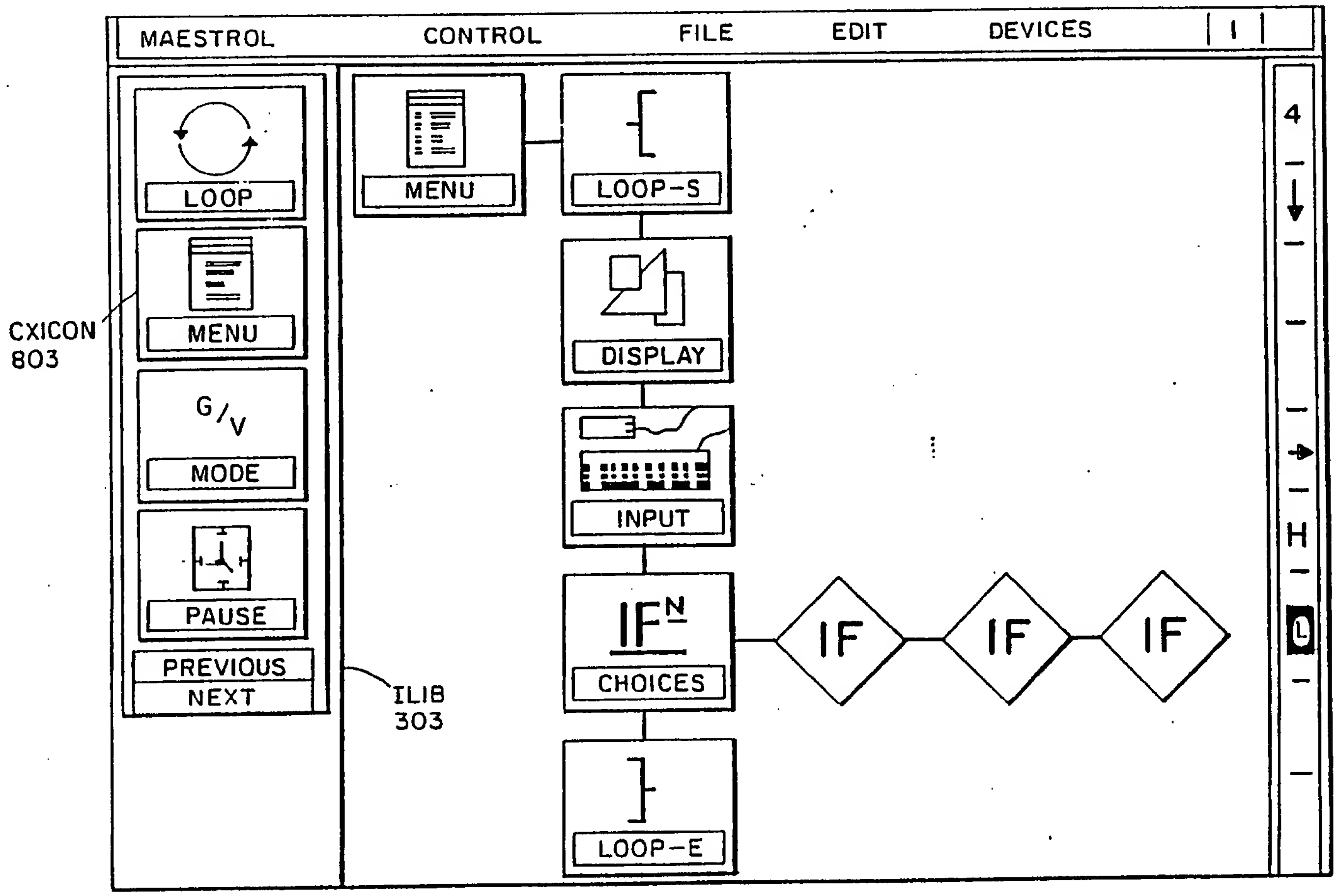


FIG. 8: ASCR(5) 801

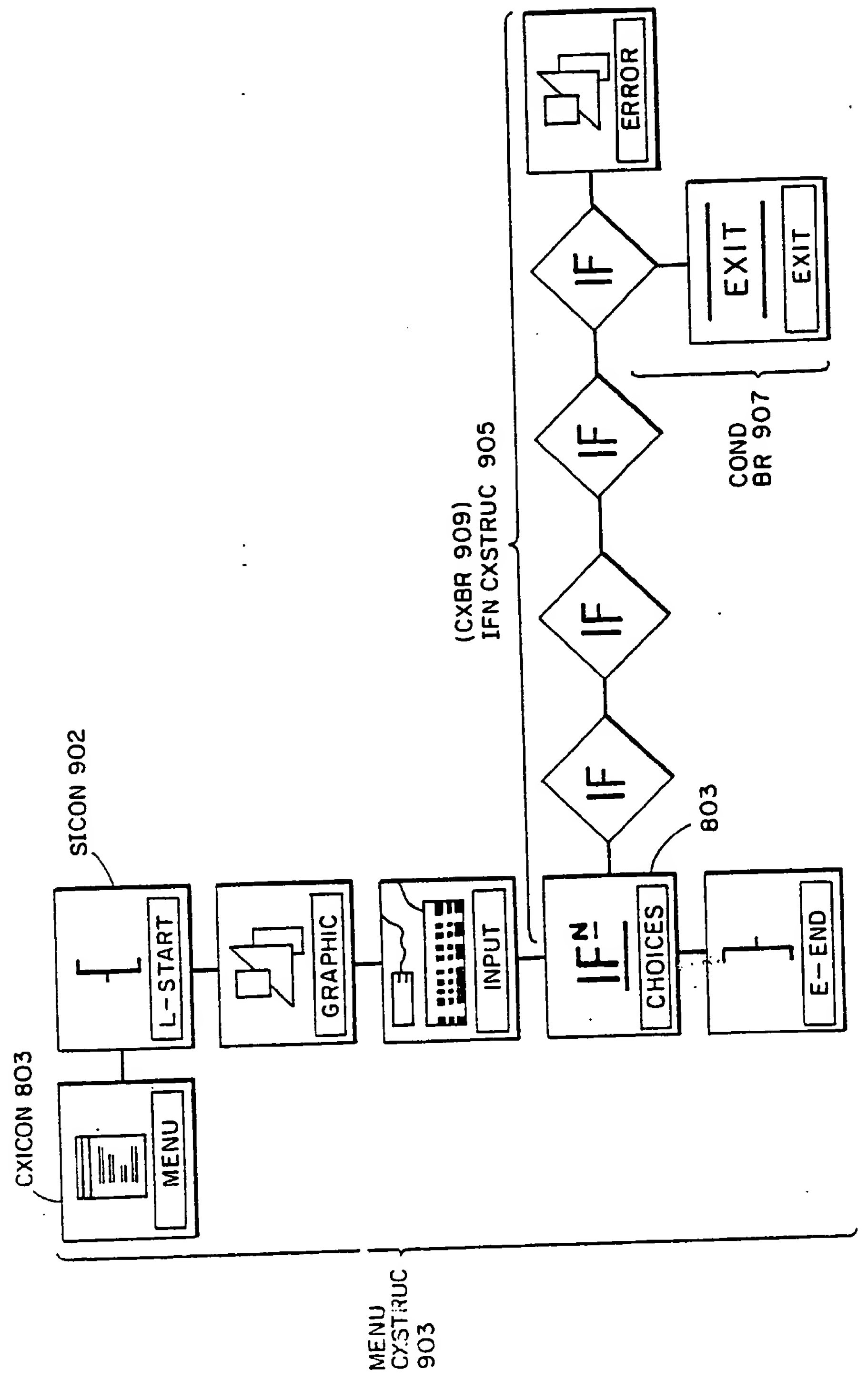


FIG. 9: CXSTRUC 901

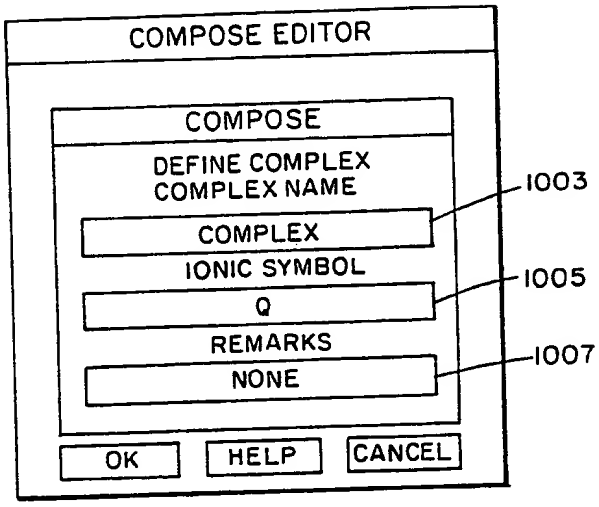


FIG. 10: COMPOSE EDITOR WINDOW 1001

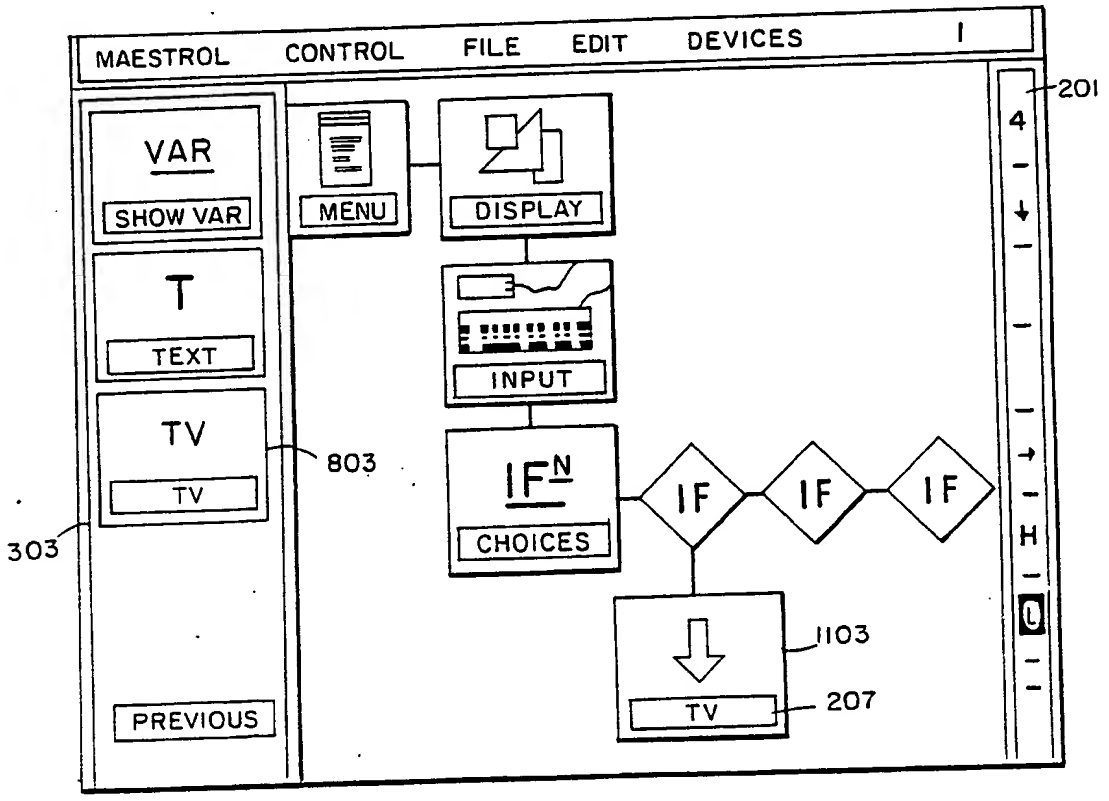


FIG. 11: ASCR(6) 1101

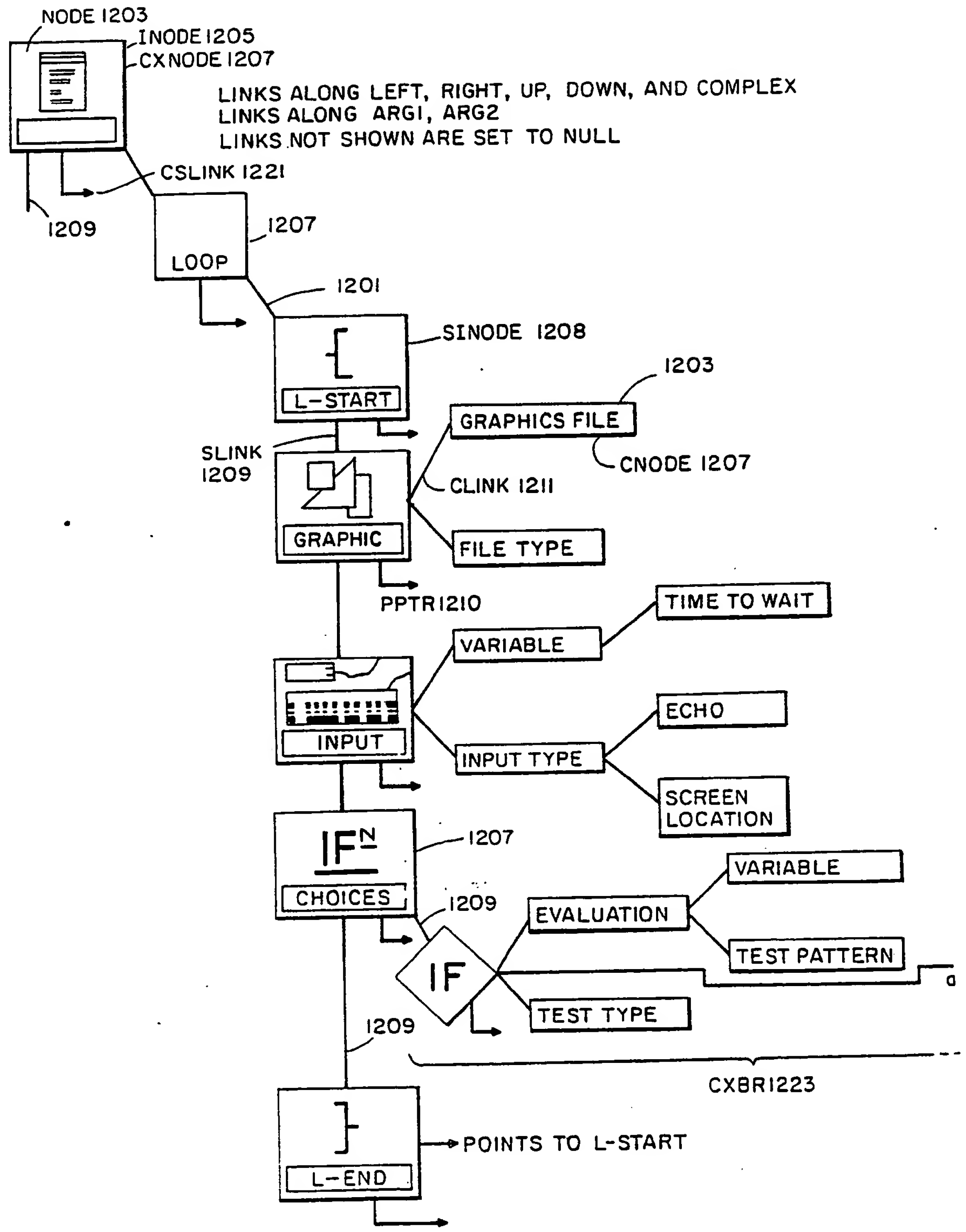


FIG. 12: PROGRAM STRUCTURE (PSTRUC) 1201

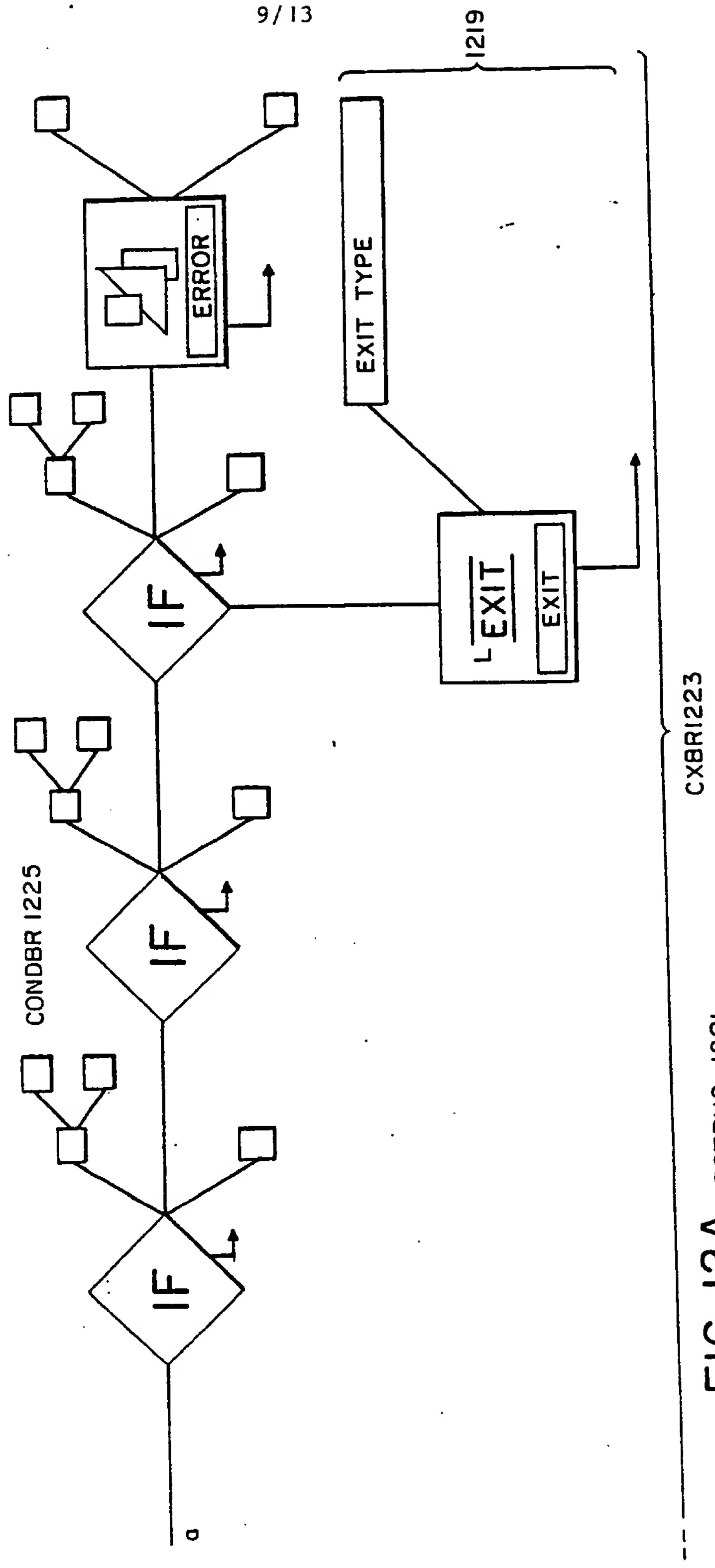


FIG. 12A: PSTRUC 1201

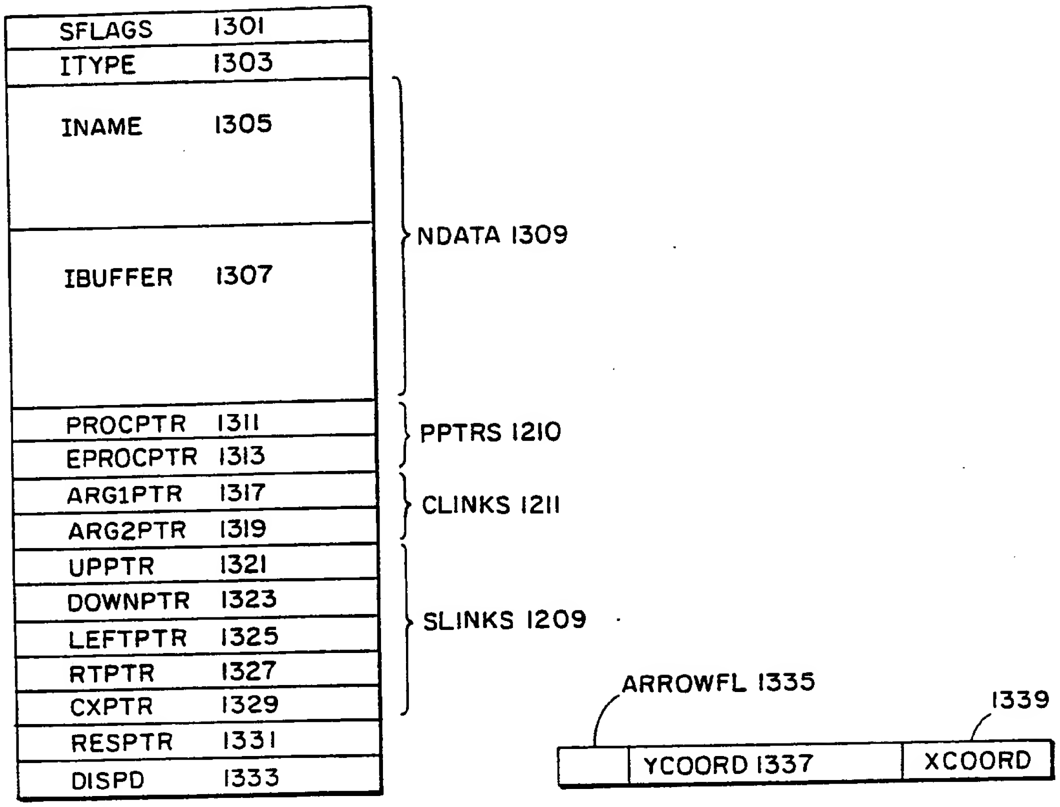


FIG. 13: DETAIL OF NODE 1203

1411

STATE FLAGS	1413
LOOP FLAGS	1415
SCR WIDTH	1417
SCR HEIGHT	1419
SCRX	1421
SCRY	1423
ICONX	1425
ICONY	1427
ISIZE	1429
ILSIZE	1430
HEAD-PTR	1431
CURR-PTR	1433
REF-PTR	1435
S-RANGE-PTR	1437
E-RANGE-PTR	1439
HOLD PTR	1443
LOOP PTR	1445
LIBBPTR	1446
LIBHPTR	1447
LIBREFPTR	1449

FIG. 14A: CONTROL VARIABLES 1411

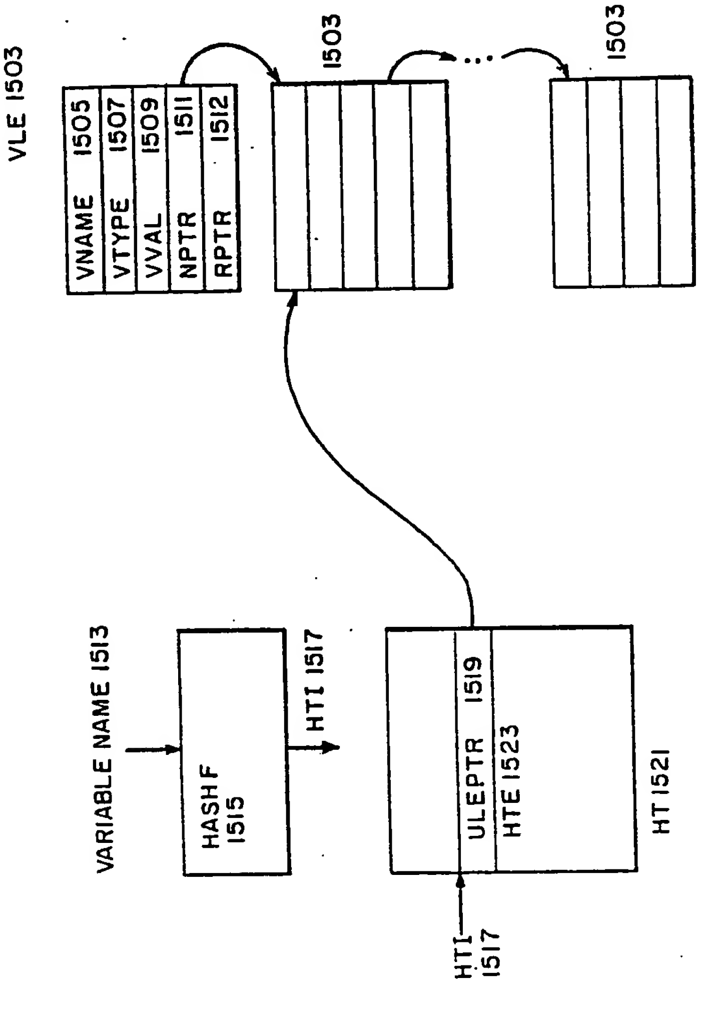


FIG. 15: VARIABLE LIST 1501

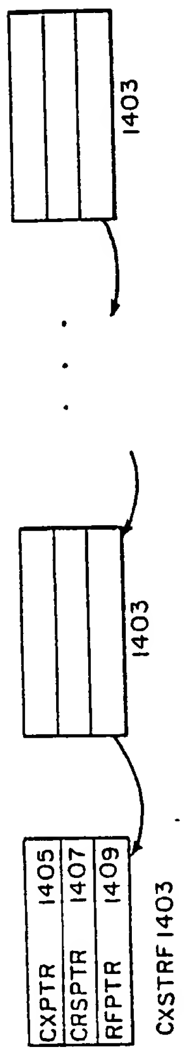


FIG. 14: COMPLEX ICON STOCK 1401

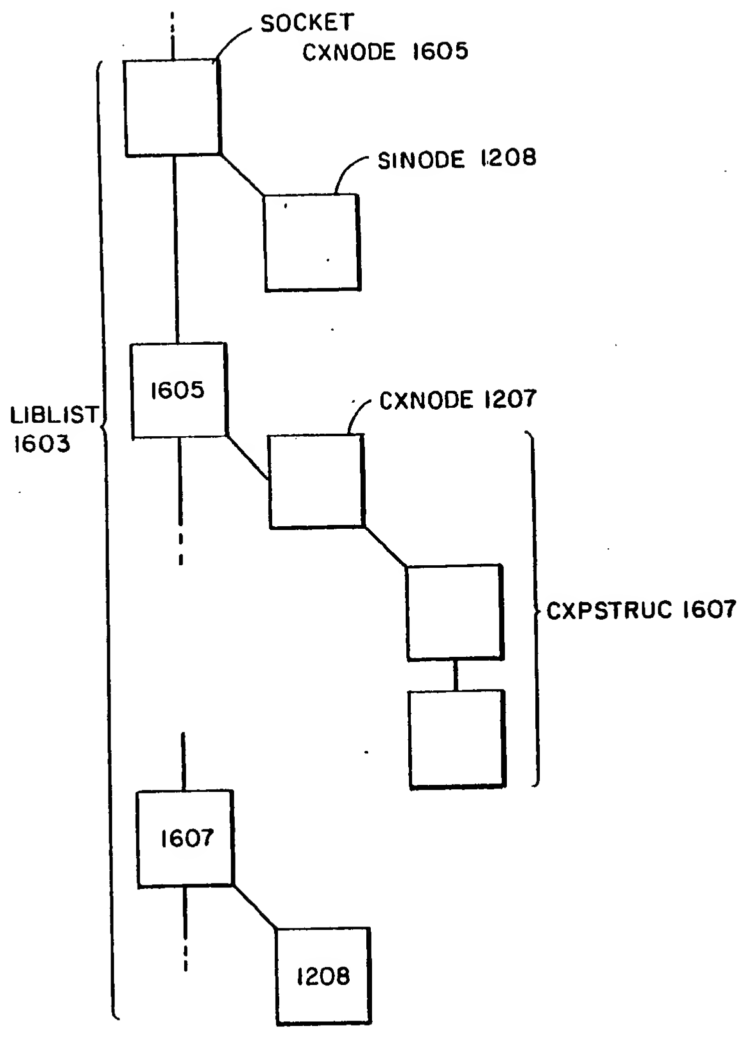


FIG. 16: LIBRARY STRUCTURE 1601

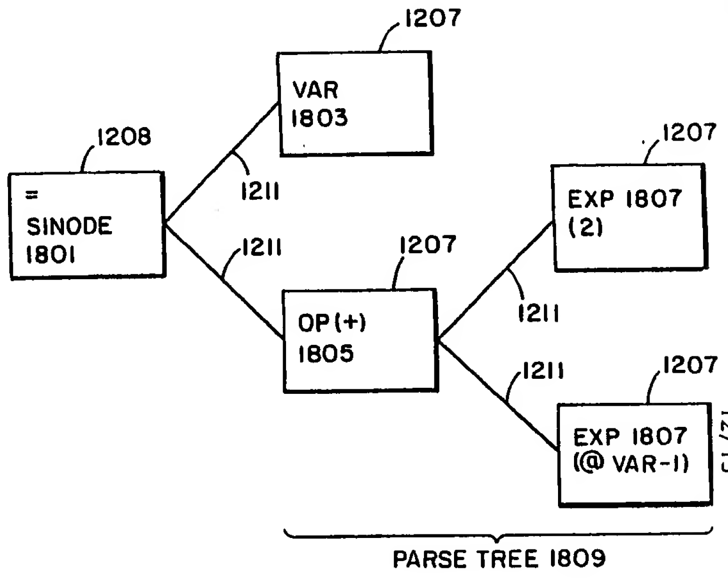


FIG. 18: ASSIGN SINODE 1801 DETAIL

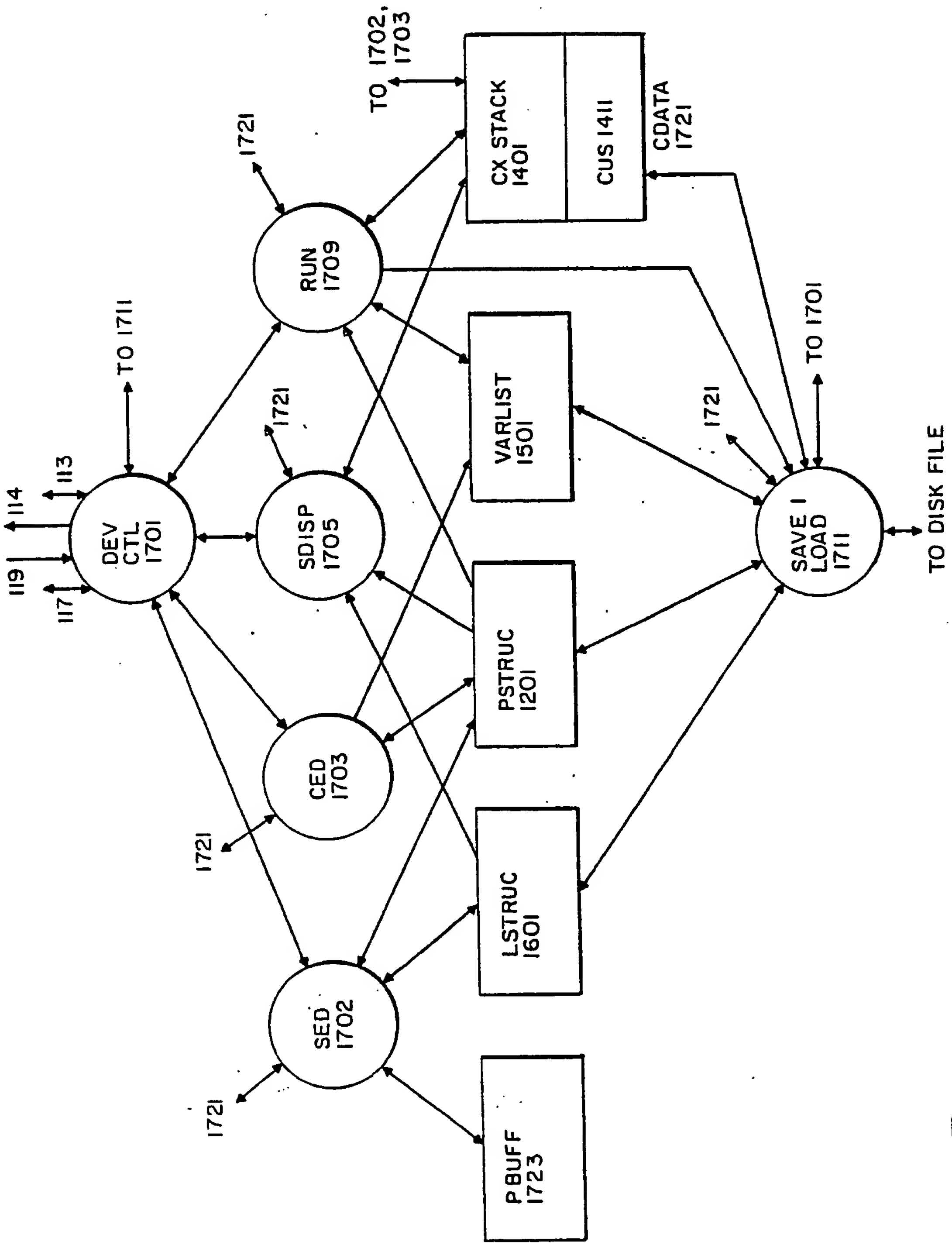


FIG. 17: PRESENTATION APPARATUS COMPONENTS

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{4} : G06F 9/44	A3	(11) International Publication Number: **WO 88/ 07719** (43) International Publication Date: 6 October 1988 (06.10.88)

(21) International Application Number: PCT/US88/01081

(22) International Filing Date: 31 March 1988 (31.03.88)

(31) Priority Application Number: 033,340

(32) Priority Date: 31 March 1987 (31.03.87)

(33) Priority Country: US

(71) Applicant: AIMTECH CORPORATION [US/US]; 77 Northeastern Boulevard, Nashua, NH 03062 (US).

(72) Inventor: OLAPURATH, John, C. ; Six Woodfield Street, Nashua, NH 03062 (US).

(74) Agents: SMITH, James, M. et al.; Hamilton, Brook, Smith & Reynolds, Two Militia Drive, Lexington, MA 02173 (US).

(81) Designated States: AT (European patent), BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent).

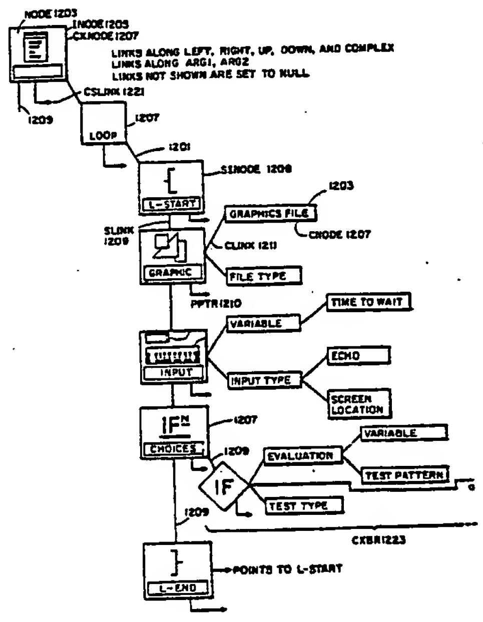
Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(88) Date of publication of the international search report: 1st December 1988 (01.12.88)

(54) Title: APPARATUS FOR ICONOGRAPHICALLY REPRESENTING AND EXECUTING A PROGRAM

(57) Abstract

A visual programming system used in a digital computer system which includes memory and a graphics display terminal. The program produced by the visual programming system is represented by a program structure in memory in which nodes represent program steps and links between the nodes the order of execution of the steps. The program represented by the program structure is executed by an interpreter component of the visual programming system. A display component of the system interprets the program structure to produce a display on the graphics terminal which represents the program as a structure of interconnected icons. The icons represent program steps and their interconnections specify the order of execution. Editing components of the system permit a program author to modify the program by manipulating icons representing the steps. The editing components include a structure editor which permits an author to add icons from an icon library to the icon structure and to move icons and copy icons already in the structure and a content editor which permits the author to add user-defined content to the program step represented by an icon. The author can further define complex icons, add them to the icon library, and use the complex icons in the program structure.



PROGRAM STRUCTURE (PSTRUC) 1201

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FR	France	ML	Mali
AU	Australia	GA	Gabon	MR	Mauritania
BB	Barbados	GB	United Kingdom	MW	Malawi
BE	Belgium	HU	Hungary	NL	Netherlands
BG	Bulgaria	IT	Italy	NO	Norway
BJ	Benin	JP	Japan	RO	Romania
BR	Brazil	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic	KR	Republic of Korea	SE	Sweden
CG	Congo	LI	Liechtenstein	SN	Senegal
CH	Switzerland	LK	Sri Lanka	SU	Soviet Union
CM	Cameroon	LU	Luxembourg	TD	Chad
DE	Germany, Federal Republic of	MC	Monaco	TG	Togo
DK	Denmark	MG	Madagascar	US	United States of America
FI	Finland				

INTERNATIONAL SEARCH REPORT

International Application No PCT/US 88/01081

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC^4: G 06 F 9/44

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
IPC^4	G 06 F 9/00

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
X	Proceedings COMPSAC 85, October 9-11, 1985 Chicago, US, IEEE (US) A, Arora et al.: "An overview of the VISE visual software development environment", pages 464-471, see page 465, right-hand column, lines 25-31; page 467, right-hand column, line 16 - page 469, right-hand column, line 39; page 470, left-hand column, lines 15-18; figures 1-4 --	1-57
X	WO, A1, 8505204 (ANALYSTS INTERNATIONAL CO.) 21 November 1985 see page 9, line 18 - page 13, line 24; claims 1,5-7; figures 3,4 --	16-22,32,36, 49-57
A	Electronics Design, vol. 34, no. 19, 21 August 1986 (Hasbrouck Heights, US) M. Schindler: "Pictures propel programming to a new plane", pages 94-104, see page 96, right-hand column, lines 19-22; figure 3 --	1-4 ./.

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search: 15th October 1988	Date of Mailing of this International Search Report: - 2. 11. 88
International Searching Authority: EUROPEAN PATENT OFFICE	Signature of Authorized Officer: [illegible] ~~P.C.G.~~ VAN DER PUTTEN

Form PCT/ISA/210 (second sheet) (January 1985)

See notes on accompanying sheet

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)

Category *	Citation of Document, with indication, where appropriate, of the relevant passages	Relevant to Claim No
A	IEEE Transactions on Software Engineering, SE-12, no. 6, June 1986, IEEE (New York, US) S.S. Yau et al.: "A survey of software design techniques", pages 713-721, see page 716, right-hand column, line 44 - page 717, left-hand column, line 41 --	1-4
A	Computer, vol. 17, no. 11, November 1984 (Long Beach, US) E. Glinert et al.: "PICT, an interactive graphical programming environment", pages 7-25 cited in the application ------	

ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO. US 8801081 SA 22458

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report.
The members are as contained in the European Patent Office EDP file on 27/10/88
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO-A- 8505204	21-11-85	AU-A- 4351185	28-11-85
		EP-A- 0180636	14-05-86
		US-A- 4742467	03-05-88

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82